UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sistema conversacional para acceso a la información de una plataforma electrónica de participación ciudadana

Autor: Carlos Ramos Mateos
Tutor: Iván Cantador Gutiérrez

JULIO 2020

Sistema conversacional para acceso a la información de una plataforma electrónica de participación ciudadana

AUTOR: Carlos Ramos Mateos
TUTOR: Iván Cantador Gutiérrez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2020

Resumen (castellano)

En este Trabajo Fin de Grado se propone diseñar y desarrollar un sistema conversacional para acceder a la diversa información existente en una plataforma de participación ciudadana, como son las propuestas e iniciativas creadas por los ciudadanos y los comentarios y debates originados sobre ellas. En particular, se plantea implementar el sistema para Decide Madrid, la plataforma electrónica de presupuestos participativos del Ayuntamiento de Madrid, y hacer uso de Google DialogFlow como tecnología de procesamiento del lenguaje natural y conversación persona-ordenador.

El sistema, de este modo, se compone de un frontend, hecho mediante la integración de un agente de Dialogflow en un bot de Telegram, a través del cual se realiza la comunicación con el usuario; y un backend, compuesto por un servicio REST de Spring Boot, que comunica y gestiona las peticiones recibidas por el agente y las resuelve consultando una base de datos PostgreSQL que almacena la información del sitio web de Decide Madrid.

Este sistema tiene como propósito el abordar una serie de problemas que actualmente poseen las plataformas electrónicas de presupuestos participativos, como la baja participación ciudadana debida a la poca amigabilidad de las interfaces y las limitaciones de los motores de búsqueda y filtrado de información de las plataformas.

Este documento comienza con los objetivos y motivaciones que han iniciado este proyecto Se seguirá con una breve introducción a los conceptos de presupuesto participativo y sistema conversacional. Se proseguirá con el estado del arte, donde se profundizará más en los antecedentes de tantos presupuestos participativos y de sistemas conversaciones. A continuación, se hablará del diseño, donde se analizará los requisitos del sistema, se justificará el porqué de la arquitectura elegida, se realizará una breve mención a la base de datos explicando su origen, su estructura y sus propiedades, acabando con una explicación del diagrama conversacional ideado. Una vez visto, se avanzará hasta la sección de desarrollo donde se hablará acerca del desarrollo del sistema conversacional como del sistema web que lo complementa. Posteriormente, se explicará el desarrollo de las pruebas y los resultados. Finalizando con un análisis del trabajo realizado y la exposición de un trabajo a futuro.

Abstract (English)

In this Bachelor Thesis, we present the design and development of a conversational system to access the heterogeneous information generated in a citizen participation e-platform, such as project proposals and initiatives created by citizens, and comments and debates originating from them. In particular, we have implemented the system for Decide Madrid, the electronic platform for participatory budgeting of Madrid City Council, using Google DialogFlow as natural language processing and person-computer conversation technology.

The system is made up of a frontend, which integrates a DialogFlow agent into a Telegram bot, which enables the communication with the user; and a backend, made up of a Spring Boot REST service that manages the requests and corresponding responses of the agent, and a PostgreSQL database that stores the data publicly available in the Decide Madrid website.

The purpose of this system is to tackle a series of problems that the electronic participatory budgeting platforms currently have, such as low citizen participation due to the unfriendliness of the interfaces, and limitations related to the platforms' search and information filtering engines.

This document begins with the objectives and motivations of the project. It provides a brief introduction to the concepts of participatory budgeting and conversational systems. It then presents a revision of the state of the art, where we go deeper into the background of some participatory budgeting and conversation systems. Next, it presents the design of the developed system, analyzing the requirements, justifying the chosen architecture, briefly describing the database origin, structure and properties, an ending with an explanation of the devised conversational diagram. It then provides a section about the development of the conversational system and the complementary web system. Later, some preliminary tests are explained. The document ends with an analysis of the work done and possible future work directions.

Palabras clave (castellano)

Presupuesto participativo, sistema conversacional, asistente personal, agente conversacional, base de datos, Dialogflow, Spring Boot, Maven, PostgreSQL, Java

Keywords (inglés)

Participatory Budgeting, conversational Systems, virtual assistant, chatbot, database, Dialogflow, Spring Boot, Maven, PostgreSQL, Java

Agradecimientos

Me gustaría dedicar este trabajo a todas aquellas personas que me han inspirado y empujado a lograr este objetivo.

Agradecer a todos aquellos profesores la paciencia que han mostrado a lo largo de estos años y toda la ayuda que me han brindado, en especial a mi tutor Iván, por no solo haberme dado la oportunidad de realizar este trabajo a pesar de todas las circunstancias que hemos vivido en el último año.

Y finalmente a mi familia, en especial a mis padres. Gracias por enseñarme que con los valores del esfuerzo y del trabajo, todo se consigue.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

En este aparatado se presentarán las motivaciones que han conducido a la realización de este Trabajo de Fin de Grado comenzando con una breve introducción a los presupuestos participativos, mencionando los beneficios que aportan, así alguno de los problemas de este tipo de plataformas.

La resolución de uno de estos problemas será el objetivo final de este Trabajo de Fin de Grado, dando como solución la utilización de un sistema conversacional, fundamentándose en las ventajas principales que suponen este tipo de tecnología.

Por último, se realizará un desglose de las diferentes secciones del documento, así como una breve explicación de los temas que se discutirán en los mismo.

1.1 Motivación

Los presupuestos participativos, Participatory Budgeting (PB) en inglés, son "procesos democráticos en que los miembros de cada comunidad deciden como gasta parte del presupuesto público. Dando capacidad de decisión sobre dinero real" (The Participatory Budgeting Project, 2020).

Los presupuestos participativos surgieron por primera vez en Puerto Alegre, Brasil, en 1989, como una medida con el objetivo de paliar la creciente pobreza debida al rápido crecimiento de la ciudad y a la falta de servicios básicos. Gracias a la plataforma se logró una mejora social cuantificable, lo que se tradujo en una mayor participación ciudadana pasando de los 1000 participantes iniciales, a los 40.000 pasados un año (Lvovna & Votto, 2018).

Desde entonces, los presupuestos participativos se han esparcido en más de 3000 ciudades alrededor del mundo tanto a diferentes niveles políticos (The Participatory Budgeting Project, 2020). Algunos ejemplos de ciudades podrían ser: Nueva York, Buenos Aires, Cambridge (Massachusetts), Reikiavik, Bangalore, Seúl, Madrid, Barcelona, entre otras muchas.

De este tipo de procesos vamos a destacar los presupuestos participativos electrónicos, e-Participatory Budgeting(ePB) en inglés, que soy aquellos que se desarrollan en un ámbito electrónico.

En estas plataformas electrónicas de presupuestos participativos de grandes ciudades hay un gran número de propuestas e iniciativas ciudadanas, por ejemplo, la media de alrededor de 6K propuestas y 21K comentarios por año en Decide Madrid[1] (Cortés-Cediel et al., 2020). Tal sobrecarga de información dificulta al usuario el acceso a información relevante, esta situación se agrava aún más debido a los sistemas actuales, ya que por lo general los motores de búsqueda de estas plataformas están limitados y basados en palabras clave.

[1] https://decide.madrid.es

Esta limitación degenera en uno de los grandes problemas de este tipo de plataformas, que es la baja tasa de participación ciudadana (p.e., la participación de Decide Madrid[2] se estima el 1% y el 1.5% del electorado), lo cual afecta a la credibilidad de tanto ciudadanos como responsables políticos acerca de la utilidad y fiabilidad de la información recogida de este tipo de plataformas.

Es por ello, que surge la necesidad del desarrollo de interfaces usuario-máquina más intuitivas que faciliten la consulta y exploración de los repositorios que solucionen este tipo de carencias a la hora de mostrar y filtrar información que pueda resultar relevante al ciudadano.

En este contexto, se introducen los sistemas conversacionales, siendo estos los sistemas que a través de los que podemos comunicarnos con lenguaje natural. En particular, se profundizará en los chatbots, siendo estos aquellos sistemas que nos permiten comunicarnos a través de una interfaz de usuario multimedia por canales de texto, de voz o mixtos, estando por lo general alojados en servicios de terceros.

Es por ello por lo que no es de extrañar que los chatbots hayan supuesto un profundo cambio en nuestra forma de comunicarnos con el mundo, y su implementación aporten numerosos beneficios, como son la completa disponibilidad, respuestas en tiempo real, un ahorro en personal, lo fácil que son de implementar en comparación con una nueva aplicación o lo fácilmente integrables que son en entornos de por sí ya muy poblados como podrían ser Facebook o Telegram.

Abordando este reto, en este Trabajo de Fin de Grado, se plantea el desarrollo de un prototipo de sistema conversacional que mediante el uso de lenguaje natural sea capaz de facilitar las consultas de los usuarios de este tipo de plataformas, procurando resolver la problemática previamente planteada. Como caso de uso y prueba de concepto del sistema utilizaremos Decide Madrid[2], la plataforma electrónica de presupuestos participativos del Ayuntamiento de Madrid.

1.2 Objetivos

El objetivo del Trabajo de Fin de Grado es el diseño y desarrollo de un prototipo de sistema conversacional a través el cual un usuario pueda consultar propuestas de una plataforma de presupuestos participativos filtrando las diferentes propuestas según sus intereses.

Con este objetivo en mente, el sistema conversacional deberá reconocer las intenciones del usuario, comunicándosela al sistema web, quien deberá buscar en la base de datos y elaborar la respuesta. En particular, se realizará un chatbot por medio de la herramienta Dialogflow[3].

[2] https://decide.madrid.es

[3] https://dialogflow.cloud.google.com

Para ello, el sistema conversacional se basará en un sistema de etiquetas, debiendo identificar de forma correcta estas para poder aplicar una serie filtros (p.e., categoría, temática o localización) que permitan acotar la búsqueda de propuestas por parte del usuario. De forma adicional, el sistema debe ser capaz de clasificar las propuestas según su grado de controversia, permitiendo acotar estas búsquedas por los filtros previamente explicados

Una vez realizadas estas consultas, se habilitará la posibilidad al usuario de realizar búsquedas concretas sobre una propuesta concreta, mostrando información según la vaya solicitando el usuario (p.e., resumen de la propuesta, número de apoyos o los comentarios de esta) guardando el hilo de la conversación hasta que realice una consulta diferente.

Se tiene que garantizar el correcto desarrollo del flujo conversacional, ya sea advirtiendo al usuario en los casos que no proporcione filtros o estos no sean correctos, en el caso de que no se identifique la intencionalidad o garantizando un tiempo de respuesta que permita el desarrollo de una conversación de una manera fluida.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** En esta sección se describirá lo que son las plataformas de presupuestos participativos electrónicos, enumerando algunos precedentes y continuaremos hablando acerca diferentes sistemas conversacionales que existen en la actualidad.

- **Diseño:** Esta sección la dividiremos en cuatro bloques iniciando por los requisitos que hemos planteado al sistema, una explicación de la arquitectura escogida, una explicación acerca de la base de datos y por último un diagrama conversación.

- **Desarrollo:** Se profundiza acerca funcionalidad implementada dividiéndola en dos bloques, el bloque perteneciente a la implementación del sistema conversacional y el bloque perteneciente a la creación del servicio web.

- **Pruebas y resultados:** Se explicarán las pruebas realizadas durante las diferentes etapas del desarrollo del proyecto, mediante el uso de herramientas como Ngrok, Postman entre otras.

- **Conclusiones y trabajo futuro:** Meditaremos sobre el resultado final del proyecto y señalaremos las posibilidades de mejora de cara a futuros trabajos.

2 Estado del arte

2.1 Presupuestos participativos electrónicos

A la hora de hacer política, siempre ha existido una relación entre gobernantes y ciudadanos, el espectro de esta relación se ha ido ampliando hasta que el ciudadano ha logrado alcanzar la toma de decisiones y el uso de los recursos públicos.

Pero en el mundo actual de inmediatez en el que vivimos, la ciudadanía exige una mayor participación en la toma de decisiones (Lorda, 2018), así como una mayor trasparencia dentro del sistema político (Ministerio de Política Territorial y Función Pública, 2020). Con sociedades y problemas cada vez más complejos, no es de extrañar que los gobiernos busquen formas de fortalecer estas vías de comunicación. ¿Qué puede hacer un gobierno para fortalecer estas relaciones? La respuesta se basa principalmente en distinguir tres niveles (OECD, 2001):

- Información: Un primer nivel que se asienta en relación unidireccional. Por lo que bien puede ser el gobierno comunicando las medidas tomadas a la ciudadanía (por medio de un boletín) o una demanda de los ciudadanos al gobierno (por medio de una protesta).

- Consultas: Un segundo nivel en el que el gobierno pregunta a los ciudadanos y recibe una realimentación de esta, aquí el ciudadano influye en la toma de decisiones, pero no participa en ellas de una forma directa.

- Participación: En este nivel el gobierno no solo consulta al ciudadano, sino que lo hace participe en la toma de decisiones de forma activa.

Este trabajo se centra en el tercer nivel, entendiendo la participación como "la facultad reconocida a los ciudadanos en participar en los procedimientos de elaboración de decisiones que afectan a intereses colectivos o a los suyos concretos" (Diccionario del español jurídico, 2020). En este punto se puede distinguir dos tipos dependiendo del medio por el que se realice esta participación: física, siendo esta aquella que se ejerce en un lugar real; o virtual, siendo esta la que se realiza en un medio electrónico.

En este contexto surgen como solución las plataformas de presupuestos participativos, como una herramienta de participación ciudadana mediante las cual la ciudadanía puede contribuir de forma democrática a decidir los proyectos a los que dirigir una parte del presupuesto público.

Los presupuestos participativos suelen tener una estructura de ciclo anual al igual que los presupuestos, este ciclo suele dividirse en una serie de etapas, tal como se describe a continuación:

- Diseño: Es la fase inicial en la que se prepara y planifica todo el proceso, por lo general, esto se realiza en una serie de reuniones iniciales con el objetivo de acordar las reglas y fechas límites por las que se regirá todo el proceso (pe., si el presupuesto se dividirá equitativamente entre los diferentes distritos, si podrán

participar todos los ciudadanos con derecho o voto o se extenderá a toda la ciudadanía, fechas límites para el resto de las fases, etc). Este proceso inicial suele estar guiado por una figura facilitadora que ayude a asentar estas bases.

- Lluvia de ideas: Una vez concluida la fase de diseño, los ciudadanos realizan varias reuniones participativas en las que se pueden exponer ideas o inquietudes y debatir en una primera instancia sobre las misma.

- Presentación de propuestas: Los voluntarios y delegados de la plataforma desarrollan las mejores ideas aquellas que tuvieron mejor acogida en la fase previa. Aunando aquellas que tienen objetivos o propósitos similares hasta lograr propuestas políticas.

- Votación: Los ciudadanos votarán las propuestas que ellos consideren más necesarias.

- Implementación: Los gobernantes toman las propuestas más votadas y son valoradas en el consistorio, y si se consideran aptas, se pasan a la agenda política para ser implementadas.

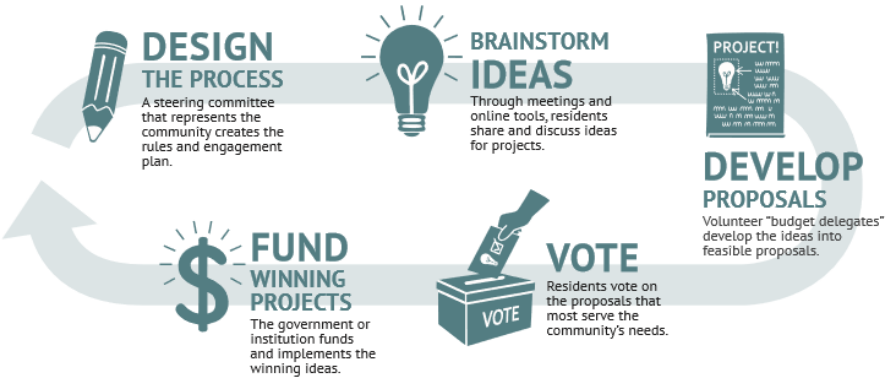

Figura 2-1: Fases principales de un presupuesto participativo [17]

La primera aparición de este tipo de plataformas se produjo en Puerto Alegre, Brasil en 1989 (de Sousa Santos, 1998), desde entonces, las plataformas de presupuestos participativos se han implementado en más de 1500 ciudades alrededor del mundo (Baiocchi & Ganuza, 2014).

A continuación, se listan algunas de estas plataformas:

- **Decide Madrid**[4]: En septiembre de 2015, el ayuntamiento de Madrid lanzó Decide Madrid. A través de esta herramienta, los residentes de Madrid pueden lanzar propuestas de proyectos e iniciativas de la ciudad con una gran variedad de temáticas, como urbanismo, transporte público, sanidad, educación o cultura. Estas propuestas pueden ser debatidas y votadas en la plataforma. Los debates en sí mismos no accionan una acción específica en el ayuntamiento, pero representan una forma útil de estimar la opinión pública. Los votos por otro lado ayudan a ver

[4] https://decide.madrid.es

el apoyo que tienen propuestas particulares. Aquellas que obtienen los suficientes apoyos son evaluadas, y si son aceptadas, llevadas a cabo por el ayuntamiento. El presupuesto asignado a las propuestas fue de cien millones en 2019 (Cortés-Cediel et al.,2020).

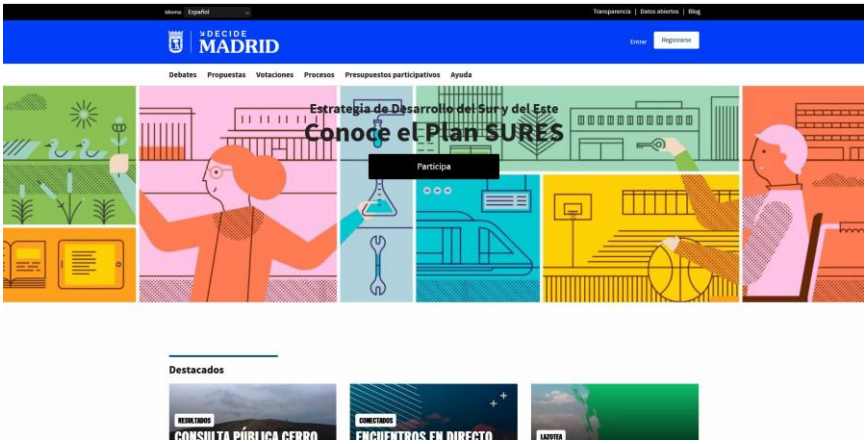

Figura 2-2: Decide Madrid

- **Dundee Decides[5]:** Una plataforma muy similar a Decide Madrid, a diferencia de esta, esta plataforma divide su presupuesto total de un 1M £ a lo largo de sus ocho distritos, lo que se traduce en 150.000 £ por distrito.

  En esta plataforma, se permite participar a cualquier residente siempre que sea mayor de once años, pudiendo tanto proponer como votar propuestas siempre que pertenezcan al distrito en el que vive (Dundee Decides, 2020):

  Este ejemplo es especialmente representativo, ya que en 2018 fue la plataforma de presupuestos participativos más utilizada de Reino Unido habiendo recibido la cantidad de 11472 votos, lo que equivale a un 10% de la población que podían participar (mayores de once años). En varias encuestas realizadas, muchos de los participantes afirmaron que esta había sido su primera experiencia en este tipo de plataformas y que estarían dispuestos a repetir para el año 2019, quedando un 80% de los encuestados contentos de la distribución del presupuesto. La plataforma logró abalar una treintena de proyectos esparcidos por toda la ciudad (PB Network, 2019).

[5] https://www.dundeedecides.org/

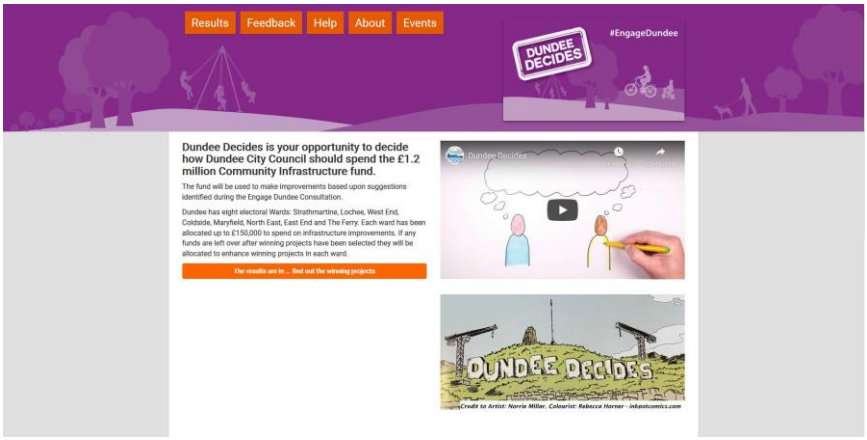

Figura 2-3: Dundee Decides

- **Better Reykjavík[6]:** Esta es una plataforma de recursos externos para la solución de problemas urbanos, fue lanzada por la Icelandic Citizens Foundation en mayo de 2010. Es una plataforma de código abierto que empezó en 2008, coincidiendo con la crisis financiera en Islandia, siendo Better Reykjavík la primera implantación exitosa de la fundación. Este ejemplo tiene una especial relevancia, ya que fue utilizada por el partido satírico "Best Party" cuando ganó las elecciones municipales del 2010, convirtiendo a la plataforma en la agenda política oficial del consistorio.

  La plataforma conecta al ayuntamiento de Reykjavik con sus ciudadanos para así mejorar las políticas y dar una mayor sensación de transparencia. La plataforma dispone de múltiples herramientas tales como la creación de la agenda política, discusiones acerca de la asignación del presupuesto, debates, externalización de contenido y propuestas, entre otras acciones políticas. Toda esta información está en manos de una innovadora IA que mejora la experiencia del usuario, así como el contenido presentado (Citizens.is, 2010).

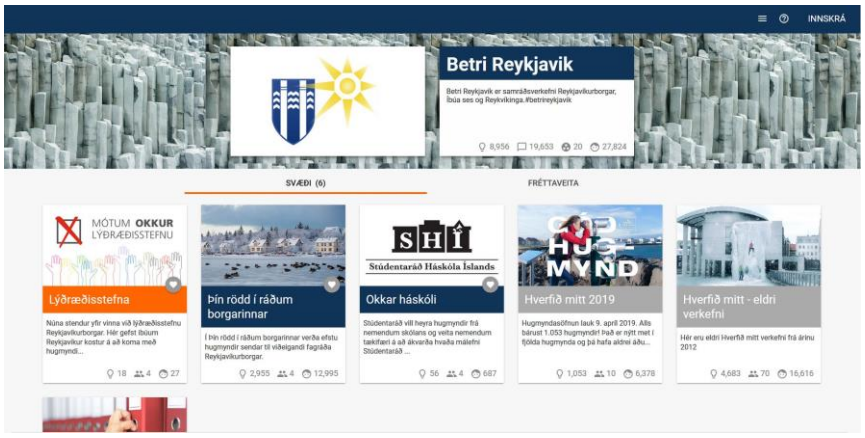

Figura 2-4: Better Reykjavík

[6] https://betrireykjavik.is/

- **New York Council Participatory Budgeting**[7]: Iniciada en 2011, permitió a los residentes decidir sobre el uso de parte del presupuesto de algo más de 35 millones de dólares, dando así voz a la comunidad. Permitiendo proponer y votar propuestas a los miembros de 33 distritos de la ciudad.

  Los presupuestos más votados son añadidos a los presupuestos e implementado por las diferentes agencias de la ciudad (New York City Council, 2020).

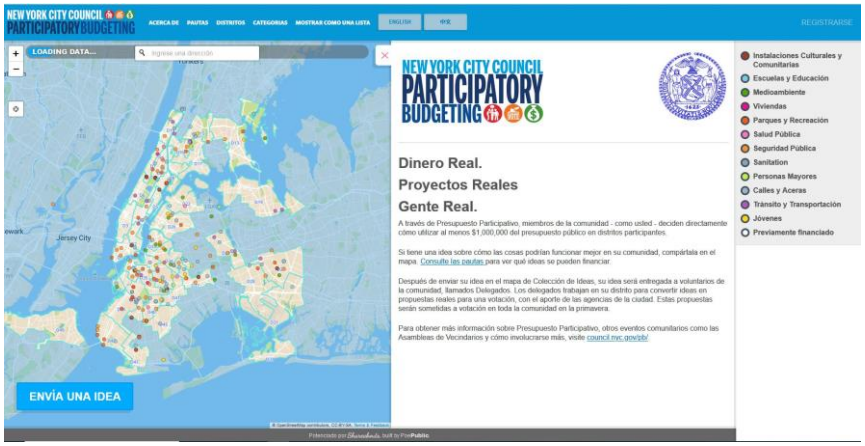

Figura 2-5: New York Council Participatory Budgeting

- **Icelandic Constitution Crowdsourcing**[8]: Otro ejemplo, a una escala de política estatal. En este caso, la constitución islandesa ha sido revisada durante este último periodo de electoral en colaboración con todos los partidos del parlamento. La universidad de Islandia & the Citizens Foundation han creado otra plataforma de comunicación entre gobierno y ciudadanos que con la intención de influir en la próxima reforma constitucional tal y como ocurrió con Better Reykjavík.

  Para 2019, alrededor de unas 39.000 personas habían visitado la página, informándose acerca del proceso de creación de una constitución. De estas personas, cabe destacar los 1100 ciudadanos que se han registrado y han aportado ideas o cuestiones a debate, lo que en un porcentaje equivaldría a un millón de personas en la formación de la constitución de los Estados Unidos, lo cual es una cantidad significativa.

  En esta ocasión, para evitar una baja implicación ciudadana, se planteó el crowdsourcing como uno de los métodos más eficaces a la hora de lograr involucrar a la ciudadanía. De forma complementaría, también se procedió al envió de una Encuesta Deliberativa (un tipo de asamblea ciudadana) seleccionado al azar alrededor de unas 300 personas de toda Islandia de una forma regular (Citizens.is, 2019).

[7] http://ideas.pbnyc.org/page/about

[8] https://stjornarskra-2019.betraisland.is/

Figura 2-6: Icelandic Constitution Crowdsourcing

Sin embargo, a pesar de estos ejemplos, este tipo de plataformas cuentan con un nivel de participación muy bajo (Zheng & Schachter, 2017) a pesar de la existencia de estudios de cómo mejorar este tipo de aplicaciones (Cantador & Cortés-Cediel, 2018).

2.2 Sistemas conversacionales

Para solucionar la problemática expuesta en el punto anterior tenemos a nuestra disposición los sistemas conversacionales.

Entendemos por sistema conversacional como: “Las aplicaciones o sistemas informáticos con los que es posible comunicarse sosteniendo una conversación en lenguaje natural, bien escrito o hablado” (Quesada Moreno et al., 2019).

Esta definición se basa en enfatizar las dos características claves que poseen dichos sistemas, siendo la primera la utilización de lenguaje natural (escrito o hablado) como base principal de la comunicación, el segundo es que la interacción se sucede en una serie de rondas en la que usuario y sistema se intercambian mensajes en lenguaje natural, dando así a una conversación. En la actualidad podemos dividir estos sistemas dentro de dos categorías: asistentes personales y chatbots.

Los asistentes y chatbots tienen varias diferencias significativas, comenzando con su finalidad, ya que mientras la funcionalidad de los asistentes personales es lo suficientemente amplia como para actuar como una especie de secretario virtual, los chatbot suelen estar mucho más acotados, limitados a resolver una serie de acciones mucho más específicas. Otra diferencia es donde están integrados, ya que mientras los asistentes suelen están entornos domóticos conocidos como *Home Speakers* o *Home Devices,* los chatbots suelen pertenecer al ámbito de páginas webs, aplicaciones de mensajería, redes sociales o aplicaciones. La última gran diferencia es quien los desarrolla, ya que los asistentes virtuales han sido desarrollados por las grandes compañías con el propósito de convertirse en nuestros asistentes personales (Hernández Á., 2018).

2.2.1 Asistentes personales

Ahora vamos a comenzar a listar a algunos de los asistentes personales más punteros del mercado actualmente:

- **Alexa** [9]: Lanzado en noviembre de 2014 por la compañía Amazon, fue inicialmente ideados para solo integrarse en los Altavoz inteligente Amazon Echo. Aunque en la actualidad, se encuentran integrados en gran variedad de dispositivos, gracias a lo cual han tenido una rápida expansión hasta convertirse en uno de los dispositivos de referencia del mercado.

  Este asistente se define por dividirse en dos rasgos principales, por un lado, tenemos los comandos de voz integrados, a través de los cuales puedes realizar una gran variedad de peticiones, y luego están las skills (que son complementos que le puedes instalar para añadirles aún más funcionalidades) existiendo actualmente más de 25.000 operativas entre las que cabe destacar las habituales que habilitan tareas de realizar llamadas o enviar mensajes, comprar productos a través de la página de la propia Amazon o hacer uso de aplicaciones de terceros y dispositivos compatibles que abarcan desde relojes inteligentes hasta electrodomésticos inteligentes (Amazon.com, Inc., 2020) (Yúbal FM, 2018).

- **Cortana** [10]: Su desarrollo comenzó en 2014 por la compañía Microsoft, con el plan de crear un asistente personal a medio-largo plazo. Actualmente se encuentra disponible en casi todos los dispositivos de Microsoft, como pueden ser Windows 10, Windows 10 Mobile, Windows Phone, Microsoft Band, Xbox One, entre otros.

  Su funcionalidad, al igual que en Alexa, se divide en comandos de voz y skills, fundamentándose principalmente el uso de aplicaciones de la propia Microsoft para la resolución de los comandos. Pudiendo, por ejemplo, consultar desde el estado del tiempo, cálculo de rutas de viaje, alarmas, recordatorios, modificaciones de agenda, realización de reservas, búsqueda por los navegadores de Edge o Bing entre otras muchas funcionalidades (Microsoft Corporation, 2020).

- **Google Assistant** [11]: Fue anunciado en 2016 por la compañía Google como parte de la aplicación de mensajería de Google Allo, estado inicialmente únicamente integrado en Google Home. Actualmente al igual que otros sistemas conversacionales ha logrado expandirse rápidamente a otros dispositivos Android, siendo una de sus peculiaridades, que al igual que previamente hizo Google Now, es capaz de detectar un input de voz como de texto (Google LLC, 2020).

  Al igual que los asistentes previos, hace uso de las aplicaciones de la compañía como las de terceros para la resolución de diferentes comandos, entre lo que podemos encontrar el realizar búsquedas en Internet, programar alarmas o agendas, ajustar la configuración de los dispositivos en los que se encuentra alojado, entre otros servicios de forma adicional, dispone de una funcionalidad de reconocimientos de información visual mediante la cámara o imágenes gracias a la tecnología de Google Lens (Google LLC, 2020).

[9] https://developer.amazon.com/es-ES/alexa

[10] https://www.cortana.es/

[11] https://assistant.google.com/intl/es_es/

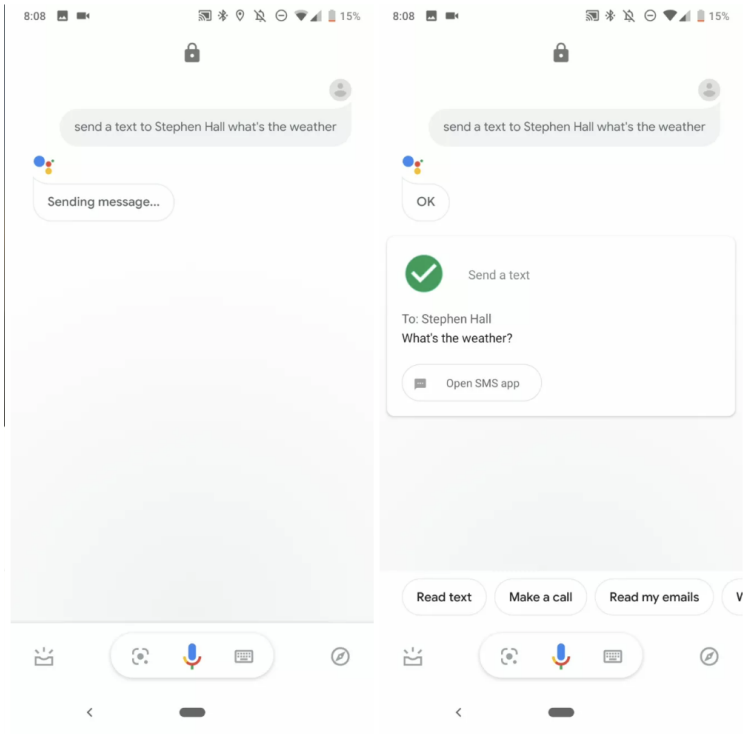

Figura 2-7: Google Assistant. Fuente: Habib Olawale, W., 2018.

- **Siri**[12]: Fue adquirida por Apple en 2010, siendo la apuesta de la compañía por los asistentes personales, estando en la actualidad integrado en la mayoría de los productos de Apple.

  Al igual que el resto de los asistentes, es capaz realizar muchas tareas como hacer llamadas o enviar mensajes, establecer alarmas, relojes, recordatorios, realizar cálculos, reservas o compras, traducciones, poner música, calcular rutas, resolver preguntas de diferentes indoles en una amplia variedad de hasta 21 idiomas. Lo más llamativo es su tecnología de inteligencia artificial, gracias a la cual puede llegar anticiparse a nuestras peticiones basándose en nuestros comportamientos previos o rutinas (Apple Inc, 2020).

2.2.2 Chatbots

Con el crecimiento exponencial del mercado de servicios móviles en la última década, han cambiado por completo nuestra forma de comunicación en este entorno, habilitando nuevas vías como podrían ser los chatbots.

Vamos a entender a un chatbot como: "una herramienta software que nos permite interaccionar con usuarios sobre una temático o en dominio de una forma natural a través de un canal de texto o voz" (Smutny & Schreiberova, 2020). Actualmente, podríamos clasificarlos según el tipo de input por el que se puede comunicar el usuario (texto, texto con botones o imágenes, mensajes de audio), en función de su aplicación práctica (mejora operativa de procesos, FAQ, servicio de atención al cliente, canal de comunicación,

[12] https://www.apple.com/es/siri/

entretenimiento, …) (Surmenok, 2016) (Planeta chatbot, 2017) o dependiendo de su arquitectura interna.

Aun así, el esquema del principio de un chatbot se puede resumir en la siguiente figura.

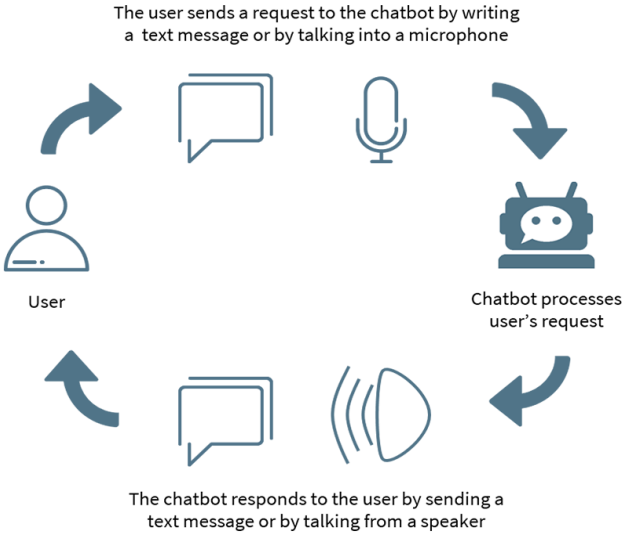

Figura 2-8: Principio de un chatbot Fuente: Synpulse, 2017.

Una vez abordado lo que son, se pasará a resumir las ventajas que proporcionan en los siguientes puntos:

- Atención al usuario personalizada: Este tipo de servicios están capacitados para ya no solo proporcionar una respuesta en tiempo real con total disponibilidad, sino que además permiten monitorizar información del usuario y recibir retroalimentación de este de una forma mucho más sencilla, lo que permite dar una atención mucho más personalizada.

- Más económico: Otro beneficio sería el ahorro de costes y tiempo, ya que es mucho más simple desarrollar un chatbot que crear una nueva aplicación multiplataforma. Además de que esto, se reduce la necesidad de personal, ya resulta más económico que contratar empleados nuevos. A esto hay que sumar la ventaja que supone tener un proceso automatizado, el cual apenas comete errores y que pueden lidiar con varios usuarios de forma simultánea.

- Fácil integración: Por último, cabe mencionar, que gracias al rápido crecimiento de las aplicaciones en las que se integran (p.e., Telegram, Facebook, Discord, etc) no es ni necesario que el usuario se instale una nueva aplicación para poder entablar una conversación con este tipo de servicios, lo que los hace mucho más accesibles (Habib Olawale, 2018).

- Open data: En un caso más particular, la existencia del Open Data habilita la existencia de sistemas informáticos basados en toda la información que la administración hace pública.

Al igual que en el apartado anterior, vamos a exponer algunos chatbots que han servido como precedente (NTS, 2020):

- Shopbot[13]: En esta ocasión se trata de un chatbot implementado a través de Facebook Messenger, el cual permite a los usuarios buscar desde artículos específicos hasta búsquedas de productos similares por medio de imágenes, de igual forma, se nos permite realizar el proceso de compra por el propio chat.

- Skyscanner[14]: Integrado tanto en Telegram como en Facebook Messenger, este chatbot ofrece sus servicios de recomendaciones de vuelos, para posteriormente poder realizar reservas desde su web.

- ImaginBank[15]: El banco móvil de Caixabank desarrolló este chatbot, el cual resuelve dudas, ofrece diferentes productos bancarios e incluso envía alertas en caso de encontrar alguna irregularidad.

[13] https://www.ebayinc.com/stories/news/say-hello-to-ebay-shopbot-beta/

[14] https://www.skyscanner.es/

[15] https://www.caixabank.es/particular/general/imagin.html

3 Diseño

Para la arquitectura del sistema, tal como se mencionó en la introducción, se dividirá en tres capas principales, siendo estas:

- Un frontend, este consistirá en un agente conversacional realizado a través de la herramienta Dialogflow que a su vez se alojará en un bot de Telegram.

- Un backend, este consistirá en un servicio web local, el cual se desarrollará en el lenguaje de programación de Java mediante el framework de Spring Boot.

- Una BD PostgreSQL situada en local que contendrá la información del sitio web Decide Madrid[16].

En conjunto, las tres capas deben recuperar las recibir las cuestiones sobre una propuesta a petición del usuario, realizar búsquedas filtrando las propuestas por los criterios especificados en la conversación y formular una respuesta que satisfaga la petición inicial del usuario.

El desarrollo se realizó siguiendo un esquema iterativo e incremental compuesto de tres interacciones, en la primera se realizó un chat conversacional con un único contexto que era identificar nombres de usuario según su identificador, en la segunda ya se implementaron contextos como la elaboración de listados de categorías, temáticas y geolocalizaciones y propuestas (sin filtros). Y en la tercera se implementó la búsqueda de propuestas bajo determinados filtros y la búsqueda de propuestas ordenadas por su valor de controversia.

A continuación, en la figura 3-1 se puede apreciar un esquema simplificado de la arquitectura de tres capas planteada:

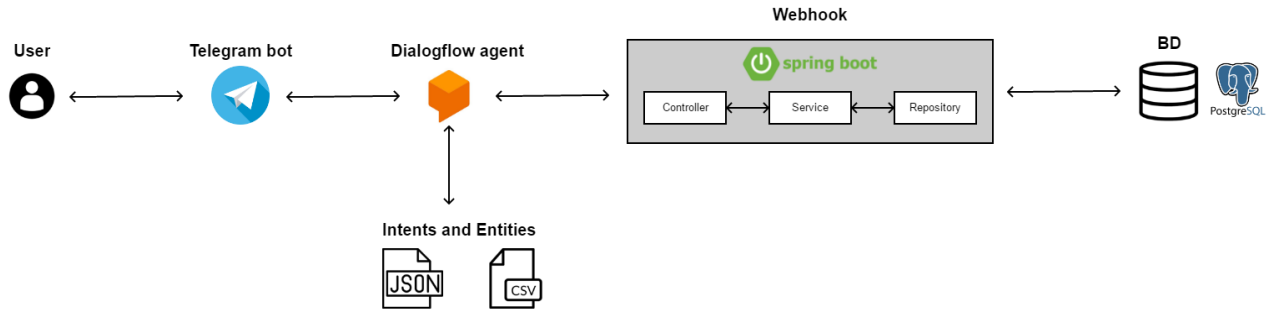

Figura 3-1: Esquema de la arquitectura

[16] https://decide.madrid.es

3.1 Requisitos

El objetivo primordial es la creación de un sistema conversacional a través del cual un usuario pueda realizar una serie de búsquedas acerca de la plataforma Decide Madrid, pudiendo reconocer y aplicar diferentes filtros a sus búsquedas con el objetivo de mejorar la herramienta de buscador que dispone actualmente la plataforma.

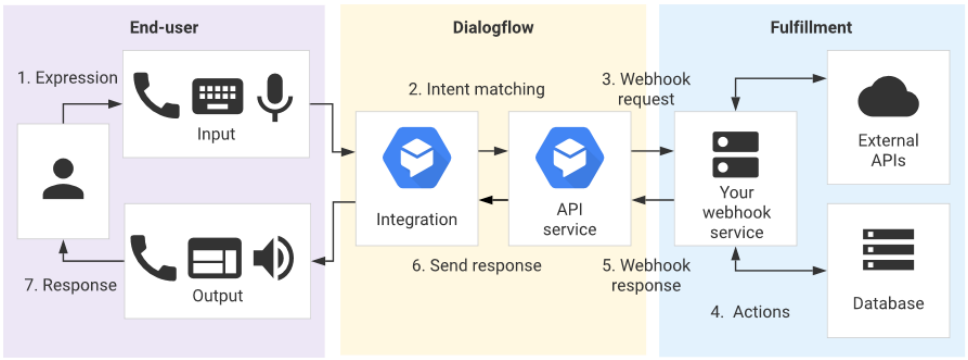

Figura 3-2: Componentes del sistema conversacional. Fuente: Google Cloud, 2020.

La funcionalidad requerida por el servicio web será la esencial. Deberá poder recibir y enviar mensajes con el agente de Dialogflow. Cuando reciba un mensaje, deberá interpretar el contexto al que pertenece, para así poder realizar una consulta en la BD y formular una respuesta acorde a la petición del agente.

Al inicio de la elaboración del proyecto, se realizó un listado de entidades que el agente debería ser capaz de reconocer a la hora de aplicar filtros de búsqueda, tales como categorías, temáticas, distritos, barrios y puntos de interés. Dichas entidades son las palabras clave que actuarán como filtro a la hora de realizar determinadas búsquedas.

De igual manera, se realizó un listado de los contextos que podrían usar esta serie de entidades, acotando la funcionalidad de nuestro prototipo de sistema conversacional debía ser capaz de comprender para poder considerarlo "finalizado".

- Ayuda: Este contexto deberá mostrar al agente una breve información sobre el chatbot, dando algunos ejemplos de cómo realizar las búsquedas.

- Listado de categorías: Deberá mostrar un listado de las categorías disponibles para facilitar las búsquedas por ese tipo de filtro.

- Listado de temáticas: Deberá mostrar un listado de las temáticas disponibles para facilitar las búsquedas que empleen este filtro. De forma opcional, se podrá acotar la búsqueda a las temáticas de una categoría en específico.

- Listado de distritos: Deberá mostrar un listado de los distritos conocidos para facilitar las búsquedas que necesiten este filtro.

- Listado de barrios: Deberá mostrar un listado de los barrios conocidos para facilitar las búsquedas que empleen este filtro. De forma opcional, se podrá acotar la búsqueda a los barrios de un distrito en específico.

- Listado de localizaciones: Deberá mostrar un listado de las localizaciones conocidas para facilitar las búsquedas que necesiten este filtro.

- Propuesta según identificador: Deberá mostrar el título de la propuesta dado un identificador de propuesta, en caso de que no exista deberá comunicárselo al usuario. El agente deberá "recordar" este contexto para poder hacer preguntas relacionadas con esta propuesta en particular, tales como:

    - *Detalle de una propuesta*: Mostrará el resumen de la propuesta y proporcionará un enlace a la página de Decide Madrid.

    - *Número de apoyos*: Informará del número de apoyos que consta la propuesta.

    - *Comentarios**: Mostrará un listado de los comentarios de la propuesta, indicando el usuario que los hizo y la fecha en la que se realizaron, así como si son una respuesta a un comentario previo.

- Propuestas filtradas según categoría o temática*: Mostrará un listado de las propuestas acotadas por una categoría o temática existente.

- Propuestas filtradas según distrito, barrio o localización*: Mostrará un listado de las propuestas acotadas por un distrito, barrio o localización conocida.

- Propuestas según la controversia: Mostrará un listado de las propuestas ordenadas de mayor o menor (o viceversa) por su valor de controversia. Esta búsqueda además debe admitir los filtros de categoría, temática o geolocalización si el usuario así lo comunica.

Los contextos marcados con *, al poder proveer mucha información como respuesta, deberán mostrar la información con un formato muy similar a PageRank, para así, no devolver unos mensaje ilegibles o saturados al usuario. Por ello, en estos contextos, el usuario puede indicar con un número a partir de ítem quiere que muestre la aplicación.

Para clarificar estos dos puntos, vamos a exponer unos ejemplos de caso de uso para ejemplificar esta funcionalidad que se espera que tenga el agente:

3.1.1 Caso de uso A: Buscar propuestas

El usuario desea buscar las propuestas asociadas a un distrito de la ciudad, para ello preguntará al agente de la siguiente manera:

"Dime las propuestas del distrito Retiro"

El agente conversacional reconocerá el cambio de contexto de la conversación, así como el distrito de interés("Retiro") y enviará una petición en formato JSON al servicio web a través de la API. Esta petición contendrá la información del contexto y el filtro de distrito adjuntado por el usuario. En caso de no haber un filtro o este no fuera válido, el agente

volverá a exigirlo hasta que se satisfaga la petición o se produzca un cambio de contexto por una nueva petición.

Una vez que se tenga la petición con toda la información, el servicio paseará esta información, lanza una búsqueda SQL a la base de datos local y generará una respuesta en formato JSON con los resultados de esta. Cuando respuesta es recibida por el agente, este la reenviará a su vez a modo de respuesta para el chatbot de Telegram.

Este caso de uso sería similar indiferentemente del filtro aplicado, ya sea temática, distrito, barrios, etc.

3.1.2 Caso de uso B: Detalle de una propuesta

El usuario ya ha buscado y ha obtenido un listado de propuestas acorde a sus intereses, donde se le muestran los títulos de las propuestas y su identificador asociado, descubre una propuesta en particular y decide preguntar por ese identificador en concreto.

"¿Qué propuesta tiene el identificador 23?"

El agente conversacional reconocerá el cambio de contexto de la conversación, así como el identificador de interés (23) y enviará una petición en formato JSON al servicio web a través de la API. Esta petición contendrá la información del contexto y el identificador de la propuesta adjuntado por el usuario. En el caso de que no se adjunte el identificador, el agente preguntará por él hasta que se le proporcioné o se cambie de contexto.

Una vez reciba el identificador, el servicio paseará la petición de forma similar que en el caso anterior y realizará una consulta SQL para comprobar si existe la propuesta, en cuyo caso se mostrará un mensaje con el título y el identificador de la propuesta seleccionada, enviando esta respuesta al agente el cual la renviará al chatbot al igual que en el ejemplo anterior.

Pero en esta ocasión, se iniciará un nuevo contexto en la conversación, el cual permite al usuario volver a continuar el contexto preguntando por:

"Háblame más acerca esta propuesta"

En este caso, el agente sabe el contexto en el que se encuentra y aún tiene presente el identificador de la propuesta que se consultó previamente. Por lo que envía una segunda petición al sistema, esta vez el servicio web realizará una nueva consulta SQL con el contexto previo, de la cual extraerá el resumen de la propuesta y adjuntará un enlace a la web donde está la propuesta. Esta respuesta es enviada al agente en formato JSON, siendo este último quien renvía el mensaje al chabot integrado en Telegram.

De forma simétrica a este caso de uso, se puede mostrar el número de apoyos o mostrar los comentarios siempre y cuando aún se conserve el contexto de la propuesta buscada.

3.1.3 Caso de uso C: Propuesta según controversia

En esta ocasión el usuario desea buscar propuestas según su nivel de controversia, por lo que realizará una consulta:

“Dime las propuestas menos controvertidas”

En este caso, el agente reconocerá el cambio de contexto de la conversación, así como el cuantificador de interés (“menos”) y enviará una petición en formato JSON al servicio web a través de la API. Esta petición contendrá la información del contexto y el cuantificador que muestra el orden que seguirá el listado de las propuestas. En el caso de que no se adjunte un cuantificador ordenará de la más a las menos controvertidas por defecto.

Una vez reciba el cuantificador, el servicio paseará la petición de forma similar que en los casos anteriores y realizará una consulta SQL ordenando las consultas según su nivel de controversia, mostrando un listado de las propuestas al igual que en el primer caso de uso.

Este caso de uso debe admitir filtros que permitan acotar la búsqueda al usuario, realizando una consulta ordenada por los criterios controversia acotada por el filtro solicitado.

3.2 Arquitectura del sistema

Antes de definirse el sistema que haría de nexo entre el agente y la base de datos, se realizó un estudio de la tecnología Spring Boot. Una herramienta que nos simplifica las tareas de creación de un proyecto Maven y el despliegue del servicio web.

Una vez finalizados los tutoriales, se pasó al diseño, el cual se dividirá en dos bloques principales, siendo el primero el encargado de establecer conexión con las BD de PostgreSQL mientras que el segundo se entregará de la integración y el intercambio de respuestas con Dialogflow.

El primer bloque se compondrá de una estructura de tres clases: entidades, repositorios y controladores.

Las entidades serán las clases ligeras encargadas de asociarse a la estructura de tablas de las bases de datos relacional, respondiendo cada entidad a un registro de la tabla. Esta clase, además, deberá de proveer los métodos GET/SET para la respectiva obtención y modificación de los campos de la tupla. Toda esta funcionalidad se implementará mediante anotaciones de Java Persistence API(JPA).

Por otro lado, los repositorios serán los encargados de exponer las operaciones CRUD que implementan toda la lógica del DAO. La implementación de estos repositorios se verá en gran parte simplificadas gracias a la extensión de repositorios de JPA.

Por último, los controladores que son las clases que implementan los métodos GET, POST y PUT mediante el uso de etiquetas de Spring y de las entidades creadas previamente.

El otro bloque es el de la integración con Dialogflow, por un lado, estará el servicio REST el cual será el encargado de recibir un mensaje y tokenizar sus parámetros, determinando cual es el contexto activo. A su vez, este bloque recibirá una respuesta de los métodos del repositorio, la renviará de vuelta al agente. Para finalizar, estará el controlador que se dividirá en una interfaz y en una clase que implementará las diferentes llamadas que harán uso de lo implementado en el primer bloque para satisfacer la consulta.

Por último, tendremos una clase principal que habilitará todos los repositorios JPA e iniciará el servicio web.

3.3 Base de datos

La base de datos de MySQL de la que partimos en este Trabajo de Fin de Grado fue generada en un trabajo previo (Cantador et al., 2020) con un *crawler* que tomó los datos a partir de la plataforma Decide Madrid[17], y del portal de Datos Abiertos[18] proporcionado por el ayuntamiento de Madrid.

Para este trabajo, se decidió hacer una simplificación de esta base datos con el objetivo de hacerla más pequeña y acta para un entorno local. Para llevar a cabo esta reducción, decidimos reducir los datos a solo aquellos pertenecientes a las propuestas del 2018. También se decidió realizar una conversión al sistema de gestión de bases de datos PostgreSQL, ya que ha sido con más se ha trabajado a lo largo de la carrera y por lo tanto mayor familiaridad se tenía.

Entidades	Original	TFG
Usuarios	17991	4853
Categorías	30	22
Temáticas	325	269
Distritos	22	22
Barrios	129	123
Puntos de interés	80	7
Propuestas	25079	4036
Comentarios	116346	8192
Propuestas con categorías	47695	3662
Propuestas con temática	55243	3662
Propuestas con localización	18731	3856

Tabla 3-1: Volcado de la base de datos

En esta implicación, también se decidió modificar la estructura para acomodarla a los requisitos, quedando la estructura reducida a los siguientes puntos:

- Los distritos se componen en un identificador único y un nombre.
- Los barrios poseen un identificador único, un nombre y están asociados a un único distrito.
- Las localizaciones (o puntos de interés) tienen un nombre y están asociados a un par de distrito y barrio únicos.
- Las categorías tienen un identificador único y un nombre.

[17] https://decide.madrid.es

[18] https://datos.madrid.es/portal/site/egob/

- Las temáticas poseen un identificador único y un nombre y una categoría asociada.

- Las propuestas por categoría tienen un trio de categoría asociada, identificador de propuesta asociada y fuente únicas.

- Las propuestas por localización siempre tienen un identificador de propuesta y distrito asociado, además pueden tener un barrio, un punto de interés y una etiqueta asociada, la combinación de todos estos parámetros será única.

- Las propuestas por temática tienen un trio de temática asociada, identificador de propuesta asociada y fuente únicas.

- Las propuestas tienen identificador único, fecha, un contador de número de apoyos y de número de comentarios, un título, un resumen de la propuesta y un enlace a la web donde esta publicada.

- Una propuesta solo puede tener una única localización.

- Una propuesta puede tener varias categorías y temáticas asociadas.

- Una propuesta puede tener entre 0 y N comentarios asociados.

- Los comentarios tienen un identificador único, un texto, una fecha, una hora y un número de apoyos positivos y negativos.

- Los comentarios tienen un usuario asociado que es su autor.

- Los comentarios pueden ser respuesta a un comentario previo, en cuyo caso tienen el identificador del comentario "padre".

- Una propuesta es realizada por un único usuario.

- Una propuesta tiene hasta cuatro métricas asociadas.

- Las métricas tienen una propuesta asociada un nombre y un valor.
    - El nombre tomará exclusivamente los valores de AGREGATION, CONVERSATION_STRUCTURE, DISCUSSION_CONTENT Y OPINION_POLARIZATION.

    - El valor de AGREGATION(controversia) tendrá un valor que oscilará entre 0 y 1.

Basándonos en estos puntos, la estructura resultante se reduce al esquema de la figura 3-3.

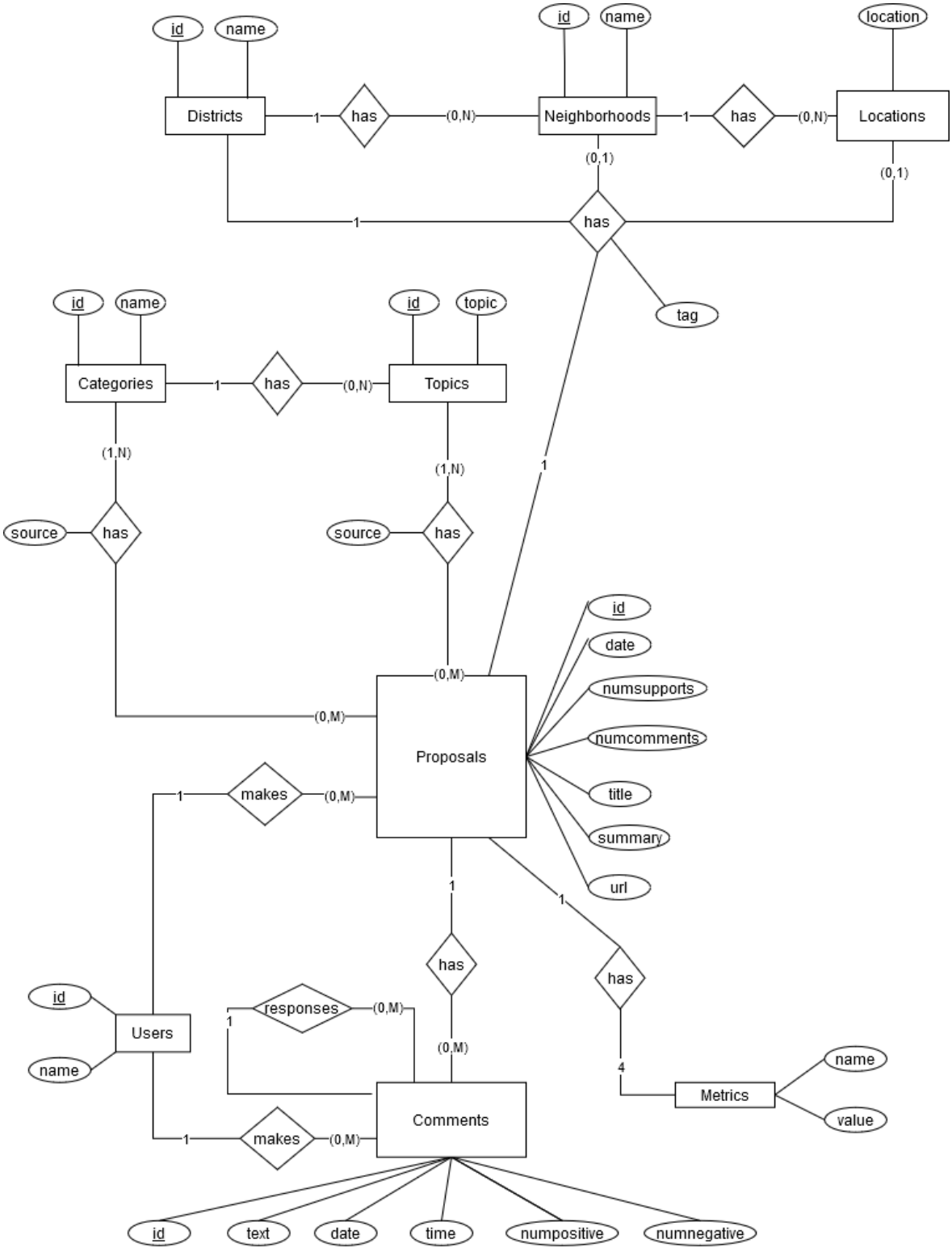

Figura 3-3: Diagrama E-R

3.4 Diagrama conversacional

Para realizar el diseño de las conversaciones se comenzó realizando un estudio de la documentación (Google Cloud, 2020) y viendo las series de tutoriales de Google: Basics of Dialogflow (Imrie-Situnayake, 2018) y Deconstructing Chatbot (Vergadia, 2019).

A continuación, se plantearon un listado de los contextos que debía ser capaz de diferenciar nuestro sistema, tal como se vio en el apartado de requisitos, y se realizó un diagrama de flujo (ver anexo a) para delimitar las posibles direcciones en las que podríamos llevar la conversación, contemplando los saltos entre diferentes contextos y las situaciones límites en los que los filtros no están presentes o no existen dentro de la plataforma.

Del primer estudio, se concluyó que los agentes de Dialogflow se dividen principalmente en dos aspectos: intenciones y entidades. Las intenciones son los contextos que debe reconocer el agente para "entender" por donde se está llevando el flujo de la conversación.

Para poder reconocerlos se basan en el reconocimiento de una serie de frases de entrenamiento. Estos contextos los podemos clasificar según su funcionalidad:

- **Auxiliares**: Estos son aquellos métodos que no necesitan una comunicación directa con la BD, por lo que solamente proporcionarán respuestas estáticas. Algunos ejemplos serían los contextos de presentación, despedida o ayuda.

- **Búsqueda**: Son aquellos métodos que sí que requieren realizar una consulta en la BD a través del servicio web, y por lo general requieren de algún tipo de entidad para actuar de filtro. Este tipo de contexto se puede subdividir a su vez de contextos según donde se deba consultar: localización, categoría, propuestas y métricas.

Para un primer prototipo, se pensaron en unas tres frases de prueba, que permitieran esbozar la funcionalidad de cada contexto, ampliándola hasta una media de una decena de frases por entidad en fases posteriores que habilitarán una mayor precisión a la hora de reconocer las intenciones del usuario.

Ejemplos de frases de entrenamiento	
Contextos	Frases
Districts from Geo_districts Intent	"¿Qué distritos tienes contemplados?"
Proposals from Proposal_categories by Category or Topic Intent	"Enumérame las propuestas de la categoría de Animales"
Text from Proposal_comments by Id Intent	"Enséñame los comentarios de esta propuesta"

Tabla 3-2: Frases de entrenamiento del agente

Respecto a las entidades, se profundizará un poco más en su diseño e intencionalidad, reflejada en los siguientes puntos:

- **Categorías**: Dialogflow debe ser capaz de interpretar palabras como “Asociaciones”, “Cultura” o compuestos como “Derechos sociales” como posibles categorías de propuestas.

- **Distritos**: Cuando un usuario escriba nombres como “Retiro”, “Latina” o compuestos como “Puente de Vallecas”, Dialogflow debe ser capaz de reconocerlos como posibles distritos de la base de datos.

- **Localizaciones**: Cuando un usuario pregunté acerca de lugares específicos como “plaza mayor”, “avenida de niza” o “feria de Madrid”, el sistema debe identificarlos como filtros de propuestas localizadas en puntos de interés.

- **Barrios**: Al igual que las otras dos entidades de localizaciones, Dialogflow interpretará nombres de barrios como “Cuatro Vientos”, “Trafalgar” o “Pilar” como filtro de búsqueda según el barrio.

- **Cuantificadores**: Debe reconocer expresiones como “más”, “menos”, “mayor” o “menor” a la hora de realizar consultas sobre métricas con el objetivo de imponer un orden en la búsqueda. Si no se especifica nada por defecto, Dialogflow asumirá que se busca de mayor a menor controversia.

- **Temáticas**: Dialogflow debe interpretar términos como “alquiler de vivienda”, “bibliotecas” o “libertades” como posibles filtros a la hora de mostrar propuestas.

4 Desarrollo

Tal como se expuso en el capítulo de diseño, se ha seguido un esquema iterativo e incremental compuesto en tres iteraciones.

En la primera iteración se realizó un agente de Dialogflow compuesto por cuatro contextos y una entidad de prueba con nombres de usuario de la aplicación. En el sistema web se desarrolló un proyecto Spring Boot que estableció las conexiones entre Dialogflow y la base de datos PostgreSQL, este sistema debía de ser capaz de recibir la petición enviada por el agente conversacional, reconocer el contexto de identificación de usuarios, realizar la búsqueda, elaborar una respuesta y enviarla al agente.

En la segunda iteración, a partir de este esquema, se expandió hasta un total de doce contextos y seis entidades, siendo las cinco añadidas los filtros de búsqueda disponibles por el usuario. En el servicio web se expandió habilitando el reconocimiento y la gestión de estos nuevos contextos, siguiendo el esquema ya implementado en la primera iteración.

En la tercera iteración se expandió hasta los veinte contextos y las siete entidades, habilitando en estos contextos todas las búsquedas relacionadas con propuestas. En el sistema web, al igual que en la interacción anterior, se habilitó la gestión de los nuevos contextos y se crearon nuevos métodos para la gestión de estas cuestiones.

A continuación, se desglosará el desarrollo del agente y el servicio web en una mayor profundidad.

4.1 Agente Dialogflow

Tal como se mencionó en la sección previa, el desarrollo del agente se ha visto facilitado por el uso de Dialogflow, el cual nos permite crear un agente conversacional con un listado de contextos conversacionales (intents) y entidades o términos clave (entities).

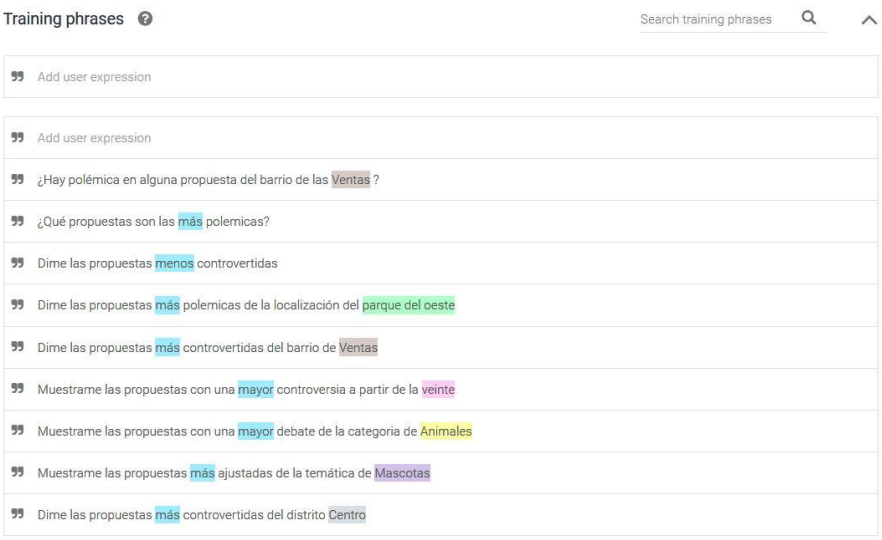

Figura 4-1. Ejemplo de frases de entrenamiento de un intent

Las intenciones se componen por unas frases de entrenamiento y unas respuestas por defecto. Pudiendo tener además una serie de parámetros reconocidos por el agente (entities) que pueden ser del sistema o personalizadas por el propio usuario, un listado de contextos tanto de entrada como de salida que permiten el traspaso de información entre intents, y una opción de fullfilment que nos permite comunicarnos con un sistema externo al agente para la elaboración de una respuesta más elaborada.

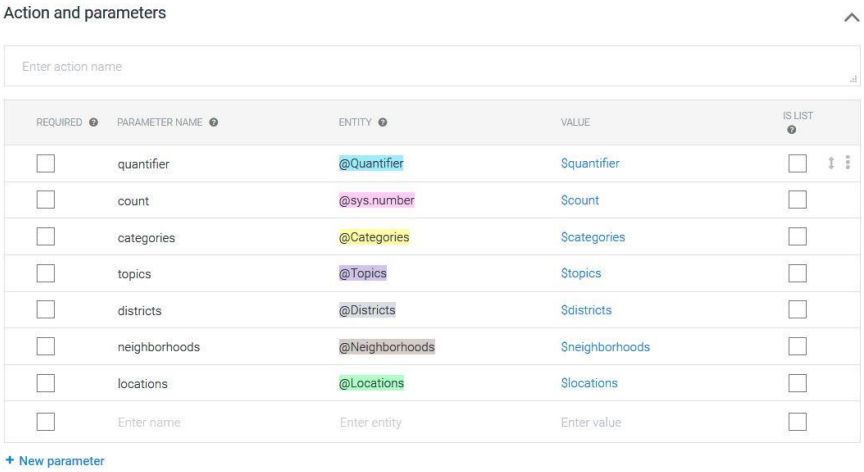

Figura 4-2: Ejemplo de parámetros dentro de un intent

En las sucesivas iteraciones se crearon los diferentes intents conforme se intentaba abarcar los requisitos del proyecto, siguiendo el siguiente desarrollo:

4.1.1 Primera iteración:

Welcome Intent:
Responde a un saludo del usuario, o al comando */start*, mostrando uno de los saludos por defecto definidos en el agente e iniciando la conversación con el mismo

Fallback Intent:
Define una serie de respuestas para el caso límite en el que el agente no reconozca la intencionalidad de la frase o esta no encaje en ninguno de los intents ya existentes.

Farewell Intent
Define una serie de despedidas para el caso en el que usuario comunique el fin de la conversación. Esta intencionalidad es el única que está definida como fin de conversación.

Users from Users by Id Intent
El usuario pregunta por un identificador dado el nombre de un usuario, el agente contacta con el servicio web y devuelve una respuesta con el nombre del identificador o un mensaje de comunicando que el usuario no existe. Si se da el caso de que el usuario preguntó por un usuario, pero no adjunté el nombre o el agente no lo reconozca como válido, el agente le preguntará de nuevo por el nombre hasta ver satisfecho el filtro o producirse un cambio de intencionalidad por parte del usuario.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

4.1.2 Segunda iteración:

Categories from Cat_categories Intent
El usuario desea información acerca de las categorías que el agente tiene contempladas, el agente realiza una petición al servicio web y devuelve una respuesta con el listado de todas las categorías que reconoce el agente.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Topics from Cat_topics Intent
El usuario solicita información acerca de las temáticas que el sistema conoce, el agente realiza una petición al servicio web y devuelve una respuesta con el listado de las temáticas que reconoce el agente, mostrando hasta un máximo de veinte. El usuario puede adjuntar un número de forma auxiliar para moverse por dentro del listado.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Topics from Cat_topics by Category Intent
El usuario pide información acerca de las temáticas relacionadas con una categoría en concreto, el agente manda una petición al servicio web y devuelve una respuesta con el listado de todas las temáticas vinculadas a la categoría adjuntada.

Si se da el caso de que el agente no reconoce la categoría, o el usuario no la ha adjuntado, el sistema conversacional solicitará el dato hasta que se le proporcione uno válido o el usuario cambio de intencionalidad con la conversación.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Districts from Geo_districts Intent
El usuario requiere información acerca de los distritos que el sistema conoce, el agente lanza una petición al servicio web y muestra una respuesta con el listado de todos los distritos reconocidos por el agente.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Neighborhoods from Geo_neigborhoods Intent
El usuario desea información acerca de los barrios que el sistema conoce, el agente envía una petición al servicio web y retorna una respuesta con el listado de todos los barrios reconocidos por el agente.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Neighborhoods from Geo_neigborhoods by District Intent
El usuario pide información acerca de los puntos de interés pertenecientes a un distrito en concreto, el agente realiza una petición al servicio web y devuelve una respuesta con el listado de todos los nombres de localizaciones que satisfacen el filtro solicitado. Si se da el caso de que el agente no reconoce el distrito o el barrio, o el usuario no la ha adjuntado ningún filtro, el sistema conversacional solicitará la información hasta que se le proporcione uno válido o el usuario cambie la intencionalidad de la conversación.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Locations from Geo_locations Intent
El usuario solicita información acerca de los puntos de interés que el sistema conoce, el agente realiza una petición al servicio web y devuelve una respuesta con el listado con el nombre de todas las localizaciones reconocidas por el agente.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Locations from Geo_locations by Distict or Neighborhood Intent
El usuario pregunta información acerca de los puntos de interés adheridos con un distrito o un barrio en concreto, el agente lanza una petición al servicio web y retorna una respuesta con el listado de todas localizaciones que satisfacen el filtro solicitado. Si se da el caso de que el agente no reconoce el distrito o el barrio, o el usuario no la ha adjuntado ningún filtro, el sistema conversacional solicitará la información hasta que se le proporcione uno válido o el usuario cambie la intencionalidad de la conversación.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

4.1.3 Tercera iteración:

Help Intent
Si el usuario solicita ayuda, el agente le muestra un mensaje definido por defecto en el que expone algunas de las funcionalidades la aplicación y expone algunas frases de ejemplo con la intención de solventar las dudas del usuario.

Proposal from Proposals by Id Intent
El usuario pregunta acerca una propuesta dando el identificador que tiene asociado, el agente realiza una consulta al sistema web y retorna una respuesta con el título de la propuesta y el identificador que tiene asociado o un mensaje de que no existe una propuesta con ese identificador, esto inicia un contexto que almacena el identificador de la propuesta para ser usado en los contextos de número de apoyos, resumen de propuesta y listado de comentarios.

En el caso de que el usuario no adjunte un identificador cuando pregunte por una propuesta concreta, el sistema conversacional se lo exigirá hasta uno válido o el usuario cambie el contexto de la conversación.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

NumSupports from Proposals by Id Intent
Una vez que el usuario ha preguntado por una propuesta en específica, puede preguntar por el número de apoyos que ha recibido la propuesta. El agente consultará tomará el identificador que tiene almacenado del contexto previo, y realizará una petición al sistema web, retornando el número de apoyos del que dispone la propuesta del contexto. Esta intención sigue conservando el contexto del identificador para cuestiones posteriores.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Summary from Proposal by Id Intent
Una vez que el usuario ha preguntado por una propuesta en específica, puede preguntar por el detalle de la propuesta. El agente consultará tomará el identificador que tiene almacenado del contexto previo, y realizará una petición al sistema web, retornando el resumen de la propuesta del contexto junto con el enlace de la propuesta en la plataforma Decide Madrid[19]. Esta intención sigue conservando el contexto del identificador para cuestiones posteriores.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Text from Proposal_comments by Id Intent
Una vez que el usuario ha preguntado por una propuesta en específica, puede preguntar por los comentarios acerca la propuesta. El agente consultará tomará el identificador que tiene almacenado del contexto previo, y realizará una petición al sistema web, retornando el listado de los cuatro comentarios de la propuesta mostrando hasta un máximo de cinco comentarios. El usuario puede adjuntar un número de forma auxiliar para moverse por dentro del listado de comentarios pudiendo acceder de esta forma a todos. Esta intención sigue conservando el contexto del identificador para cuestiones posteriores.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

Proposals from Proposal_categories by Category or Topic Intent
El usuario pide información acerca de las propuestas relacionadas con una categoría o temática en concreto, el agente realiza una petición al servicio web y devuelve una respuesta con el listado con diez de las propuestas vinculadas al filtro adjuntado o un mensaje negativo en el caso de que no existan propuestas con esos filtros. El usuario puede adjuntar un número de forma auxiliar para moverse por dentro del listado pudiendo acceder de esta forma a todos de forma muy similar a un “PageRank”.

[19] https://decide.madrid.es

En caso de que el usuario no adjunte ninguna categoría válida, el sistema web le responderá con una advertencia de que no ha proporcionado información suficiente como para procesar la petición.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

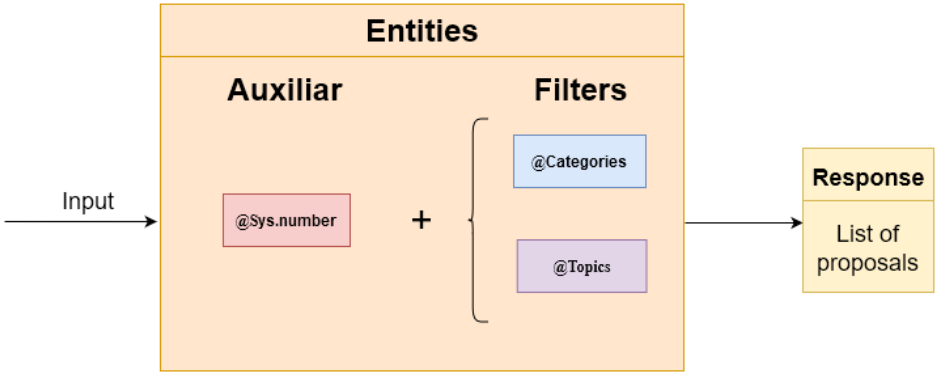

Figura 4-3: Esquema de Proposals from Proposal_categories by Category or Topic Intent

Proposals from Proposal_locations by District or Neighborhood or Location Intent
El usuario pide información acerca de las propuestas relacionadas con un distrito, barrio o punto de interés en concreto, el agente realiza una petición al servicio web y devuelve una respuesta con el listado con diez de las propuestas vinculadas al filtro adjuntado o un mensaje negativo en el caso de que no existan propuestas con esos filtros. El usuario puede adjuntar un número de forma auxiliar para moverse por dentro del listado pudiendo acceder de esta forma a todos de forma muy similar a un "Pagerank".

En caso de que el usuario no adjunte ninguna categoría válida, el sistema web le responderá con una advertencia de que no ha proporcionado información suficiente como para procesar la petición.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

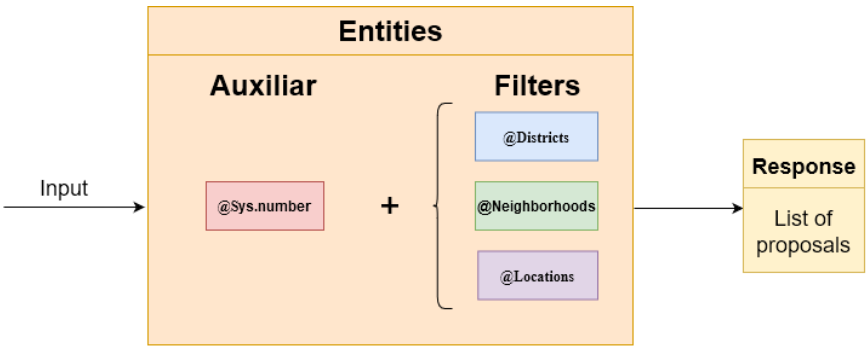

Figura 4-4: Proposals from Proposal_locations by District or Neighborhood or Location Intent

Proposal from Metrics_controversy Intent
El usuario pide información acerca de las propuestas relacionadas con un distrito, barrio o punto de interés en concreto, el agente realiza una petición al servicio web y devuelve una respuesta con el listado con diez de las propuestas vinculadas al filtro adjuntado o un mensaje negativo en el caso de que no existan propuestas con esos filtros. El usuario puede adjuntar un número de forma auxiliar para moverse por dentro del listado pudiendo acceder de esta forma a todos de forma muy similar a un "Pagerank".

En caso de que el usuario no adjunte ninguna categoría válida, el sistema web le responderá con una advertencia de que no ha proporcionado información suficiente como para procesar la petición.

En el caso de que el servidor web este caído, el agente comunicará un mensaje al usuario advirtiéndole del error.

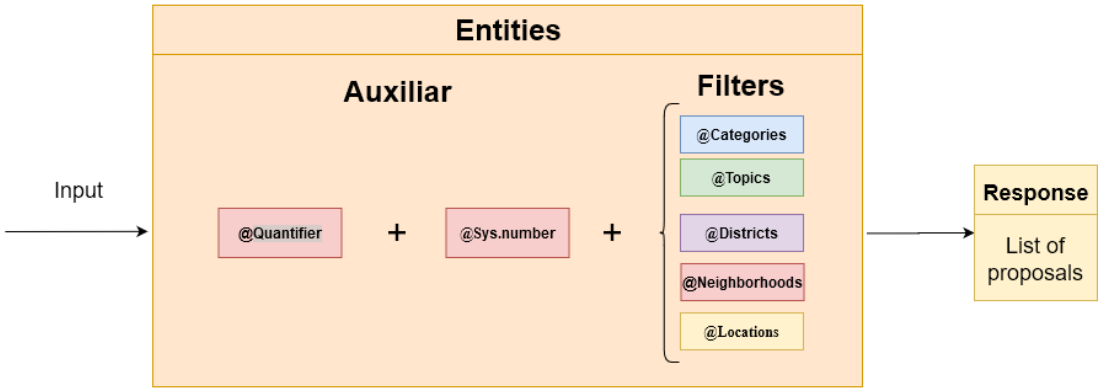

Figura 4-5: Proposal from Metrics_controversy Intent

4.1.4 Integración con Telegram

Como último paso, se integró el agente de Dialogflow con un Bot de Telegram, para ello se creo una cuenta en dicha red de mensajería. Tras esto se accedió a la interfaz web de botfather[20] y siguiendo las instrucciones del tutorial (Google Cloud, 2020) se logró un token de acceso, el cual se vinculó al agente de Dialogflow por su pestaña de Integration.

[20] https://telegram.me/botfather

4.2 Servicio web

Tal como mencionamos en el punto anterior, el servicio web tenía un diseño que se dividía en dos bloques, el primer bloque sería el DAO que permite la comunicación del servicio web con la BD y el segundo sería un modelo REST para el fullfilment con Dialogflow.

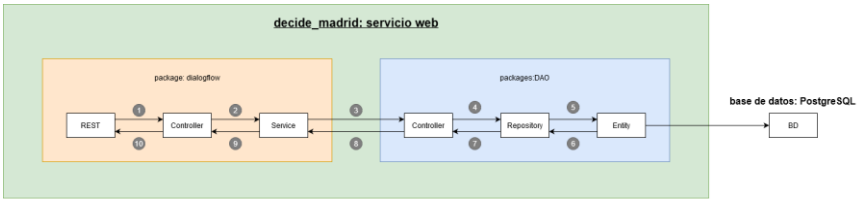

Figura 4-6: Esquema completo de la arquitectura del servicio web

Para empezar, se inició con la creación de los paquetes pertenecientes que dividían estos dos bloques, al primero bloque se dividió en un único paquete (dialogflow), en cambio al segundo, se le dividió en tres paquetes (entities, repositories y controllers) que contenían respectivamente las clases de usuario siguiendo el diseño de entidad-repositorio-controlador explicado en la arquitectura. En iteraciones posteriores se decidió la subdivisión de paquetes para separar las tablas BD con una temática similar, dando así a la siguiente división de paquetes final.

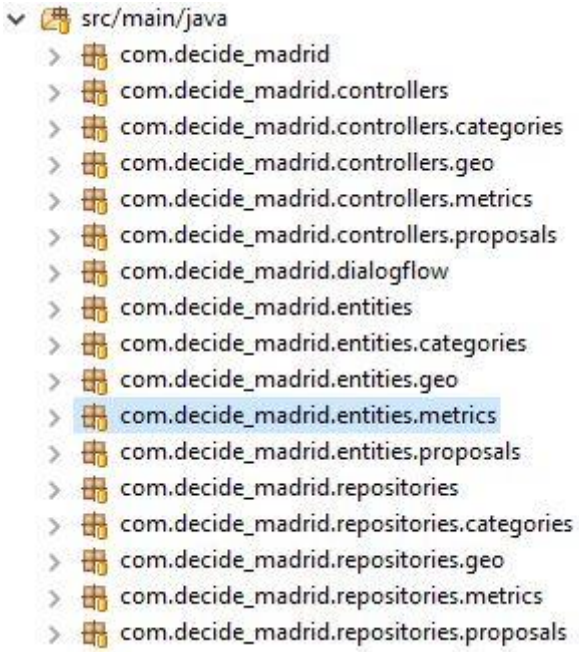

Figura 4-7: Paquetes del servicio web

Para el primer bloque, se realizó una primera versión del paquete dialogflow donde siguiendo la estructura REST-Controlador-Service se implementa el tráfico de mensajes y la resolución de peticiones con el agente y la base de datos respectivamente.

La clase REST se consta de un único método el cual recibe un mensaje JSON, lo convierte a un objecto de la clase *GoogleCloudDialogflowV2WebhookRequest*, una vez hecho esto, esta nueva clase es filtrada a través de un condicional, el cual determinará la intención de

la petición, llamando al método pertinente dependiendo del contexto (1). Este método devolverá una cadena que es la respuesta (10). Por último, la clase creará un objecto *GoogleCloudDialogflowV2WebhookResponse* y se lo enviará de vuelta al agente mediante un mensaje JSON.

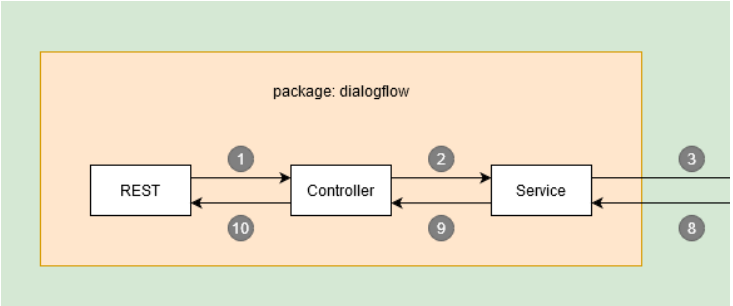

Figura 4-8: Arquitectura del bloque de integración con dialogflow

En el controlador es una interfaz donde se declaran los diferentes métodos que son necesarios para la elaboración de la respuesta (2).

Estos métodos son implementados en la clase Service, la cual contiene todos los controladores de los métodos asociados a la estructura de tablas, por lo que puede acceder a las llamadas CRUD que implementan la lógica DAO para realizar consultas SQL a la base de datos, la clase se encarga de analizar sintácticamente los parámetros dependiendo del método llamado, realizar e interpretar los resultados de estas consultas SQL (3) y en última instancia, recibir la información de las entidades (7 y 8) y elaborar una respuesta de texto la cual retorna a la clase REST (9). Tal como se puede ver reflejado en la figura previa.

Para el segundo bloque, su implementación se ha realizado siguiendo el siguiente orden: primero se definía la entidad correspondiente que se relacionaba por medio de las etiquetas de JPA y se definen los métodos GET/SET que nos permitirán sacar información de las entidades, a continuación, se creaba su repositorio extendiendo de la clase *JpaRespository* de Spring donde se definían todas las funciones CRUD necesarias para resolver las diversas peticiones que podía realizar el agente sobre la tabla, estas peticiones siempre devolverán datos del tipo de Entidad que tiene asociada. Finalmente se implementaba el controlador REST donde se definen los métodos POST, PUT y GET, y métodos auxiliares que dan acceso a las consultas SQL del repositorio.

El controlador recibe las llamadas del Servicio del primer bloque (3) y llama a los métodos CRUD del repositorio (5) debatiéndonos una Entidad, a esta entidad le realizamos llamadas a sus métodos GET/SET para extraer la información (6), la cual se pasa de vuelta al Service (7 y 8) para elaborar un mensaje.

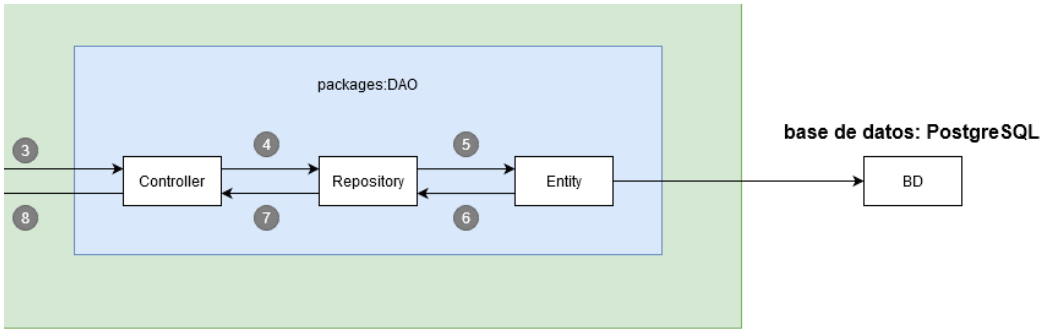

Figura 4-9: Arquitectura del bloque de integración con PostgreSQL

5 Integración, pruebas y resultados

A lo largo de las tres iteraciones hemos ido realizando diversas pruebas para comprobar el progresivo avance de cada propuesta.

Cabe mencionar que para finalizar la integración entre el agente de dialogflow y el servicio web, hace falta que el servicio se encuentre desplegado en URL pública. Puesto que todo el desarrollo tanto de la base de datos como del servicio web se ha probado en un entorno local no hay una forma directa de poder realizar unas pruebas. Para solventar este punto, usamos la herramienta ngrok[21], la cual nos permite crear una URL dinámica mediante la cual podemos exponer nuestro servicio web local, por lo con esta nueva URL pública finalmente podemos integrar el servicio web en el agente y realizar las pruebas. En nuestro caso expusimos nuestro localhost:8080 mediante el comando *ngrok http 8080*

Otra herramienta usada fue Postman[22], la cual es una aplicación que habilita el envío de peticiones HTTP REST, esto lo utilizamos en un principio para probar el correcto funcionamiento del servicio web.

5.1 Primera y segunda iteración

Para tanto la primera y segunda iteración, nos hemos limitado a realizar dos tipos de pruebas, la primera de ellas era iniciando diversas conversaciones con el agente mediante la consola que habilita el propio Dialogflow.

El objetivo de estas pruebas se podía definir en la siguiente serie de puntos:

- Ver si el agente es capaz de reconocer correctamente las intenciones del usuario con frases similares a las esperables por un usuario.

- El agente era capaz de detectar diferentes entidades o filtros, con el propósito de corregir las carencias que pudiera haber.

- Controlar que ocurría frente a diversas situaciones límite, como frases con intencionalidad desconocidas, frases incompletas sin un determinado filtro necesario para su contexto o frases con filtros desconocidos para el agente.

- Controlar la información que devuelve el sistema web comparándola con la información devueltas por las consultas SQL equivalentes realizadas a través de PgAdmin4.

El segundo de este tipo de pruebas fueron mediante la herramienta Postman. Esto lo haremos con el propósito de probar la funcionalidad del servicio web, probando en

[21] https://ngrok.com/

[22] https://www.postman.com/

particular las llamadas de los métodos POST, PUT, GET implementados en los controladores de las diferentes entities asociadas al BD.

5.2 Tercera iteración

Se continuaron realizando las mismas pruebas mismas pruebas que en las fases anteriores, aplicándoselas a lo desarrollado durante estas iteraciones. A diferencias de las dos anteriores, se volvieron a realizar pruebas similares, con la diferencia de que esta vez se harán a través del chat de Telegram.

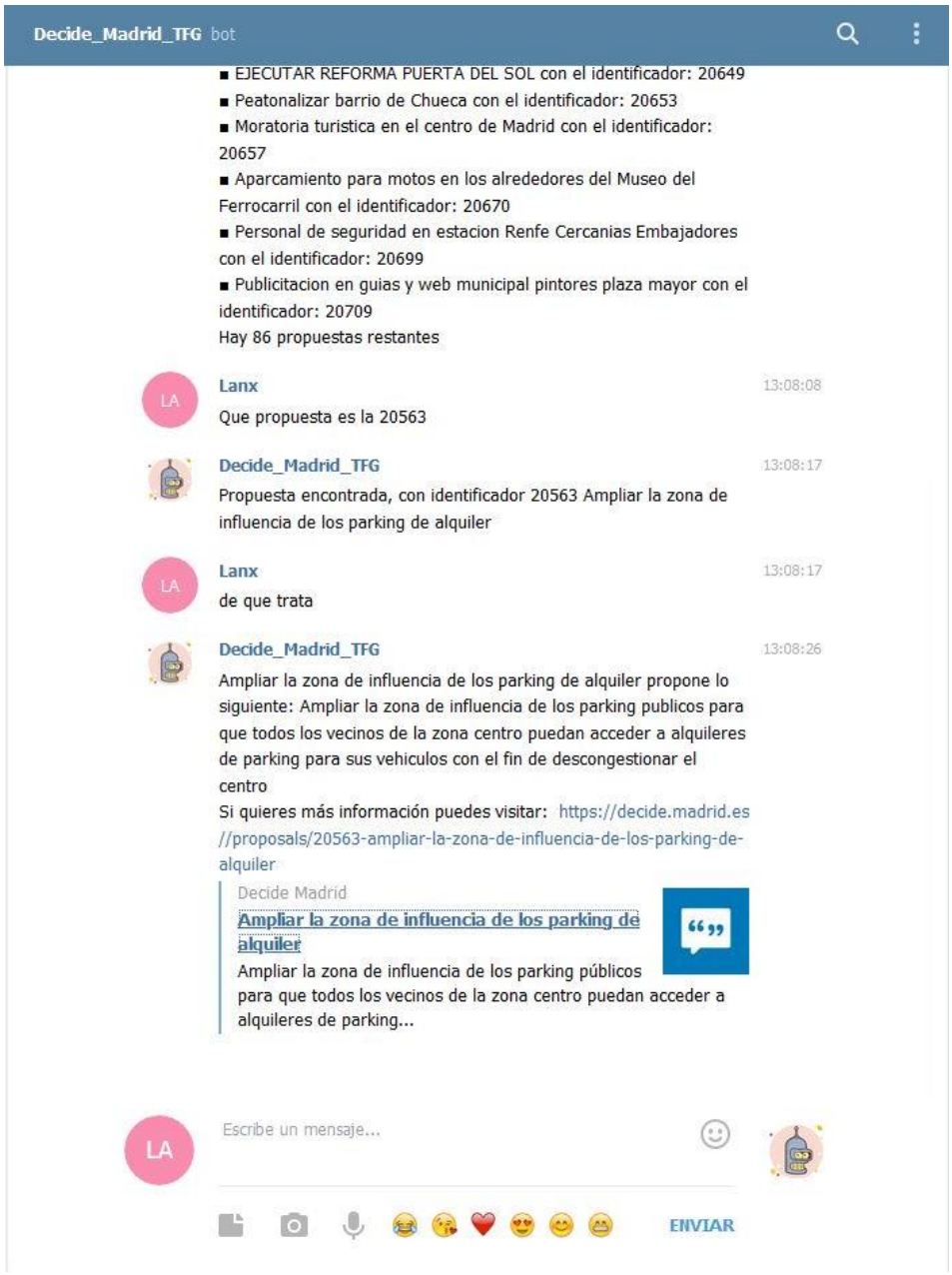

Figura 5-1. Ejemplo de prueba de conversación por Telegram

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El Trabajo de Fin de Grado ha tenido cierta complejidad debido a la gran variedad de herramientas que involucradas en su desarrollo, de las cuales no se había trabajado en asignaturas previas en la carrera, tales como Spring Boot, Dialogflow para el desarrollo, o ngrok o Postman para las pruebas.

La mayor dificultad del proyecto residía en la integración de tres servicios diferentes, por un lado, la integración del agente Dialogflow con el servicio web y la integración del servicio web con la base de datos, aparte de hacer una integración funcional, hay que recordar que a la hora de implementar un sistema conversacional es importante que los métodos sean eficientes en cuanto a tiempo para poder dar una respuesta en lapsos cortos.

El resultado final del sistema ha sido satisfactorio, puesto que el software desarrollado cumple con todos los requisitos esenciales que se habían planteado inicialmente para un prototipo inicial, que, además, resuelve un problema existente en las plataformas de participación ciudadana. Como último comentario, cabe mencionar, que se han trabajado con tecnologías relativamente nuevas que apuntan a volver punteras en un futuro.

6.2 Trabajo futuro

A lo largo del proyecto se plantearon ciertos puntos que si bien no se consideraron para este prototipo inicial serían interesantes de tratar en un futuro:

- Permitir realizar búsquedas con combinación de filtros más elaboradas, como podría ser buscar las propuestas de una categoría/ o temática particular en un distrito, barrio o punto de interés concreto.

- Ampliar los intents para tener un espectro de conversación más amplio, como, por ejemplo: propuestas con mayor número de comentarios, ordenar las propuestas de más recientes a más antiguas, propuestas que están cerca de cumplir los votos necesarios. Así como ampliar el número de usadas entities (pe., incluyendo las de @sys.date para la gestión de un filtro de fechas) para que el agente sea capaz de reconocer un mayor número de términos. Recordemos que actualmente esta limitado a los términos pertenecientes a las propuestas de 2018.

- Mejorar los flujos de conversación actuales para recibir feedback tras cada consulta, tratando de hacer una conversación más fluida y no tan “mecánica”.

- Introducir un sistema de login, para habilitar a los usuarios registrados nuevas opciones como las de proponer nuevas propuestas, editar sus propuestas planteadas, comentar y votar desde la propia interfaz conversacional.

- Añadir un sistema de recomendación que permita realizar recomendaciones mucho más exactas en base a un usuario, o que nos permitiera realizar valoraciones objetivas de las propuestas que pueden resultar más relevantes de las que no.

- Migrar el servicio web de un entorno local a un entorno en la nube, tal como podría ser Heroku o Azure. Esto se intentó realizar en una etapa final del desarrollo, pero no se logró debido a que no se encontró un servicio gratuito en PostgreSQL capaz de aguatar el tamaño de la base de datos.

Referencias

[1] Amazon.com, Inc., (2020). "Alexa and Alexa Device FAQs" [online] https://www.amazon.com/gp/help/customer/display.html/ref=hp_left_v4_sib?ie=UTF8&nodeId=201602230 [Accedido el 28 de junio de 2020]

[2] Apple, Inc., (2020). "Antes de que digas nada, Siri hace más que nunca" [online] https://www.apple.com/es/siri/ [Accedido el 28 de junio de 2020]

[3] Baiocchi, G., & Ganuza, E. (2014). Participatory Budgeting as if Emancipation Mattered. Politics & Society, 42(1), 29–50. https://doi.org/10.1177/0032329213512978

[4] Betri Reykjavik, (2020). "Betri Reykjavik" [online] https://betrireykjavik.is/domain/1 [Accedido el 27 de junio de 2020]

[5] Cantador, I., Cortés-Cediel, M.E., (2018). "Towards increasing citizen engagement in participatory budgeting digital tools, in: Proceedings of the 19th International Conference on Digital Government Research", ACM. pp. 91:1–91:2. doi:10.1145/3209281.3209389.

[6] Cantador, I., Bellogín, A., Cortés-Cediel, M. E., Gil, O. (2019) "Personalized recommendations in e-participation: Offline experiments for the 'Decide Madrid' platform", Information Retrieval.

[7] Citizens.is, (2010). "Better Reykjavík" [online] https://www.citizens.is/portfolio_page/better_reykjavik/ [Accedido el 27 de junio de 2020]

[8] Citizens.is, (2019). "Icelandic Constitution Crowdsourcing" Citizens.is. [online] https://www.citizens.is/portfolio_page/icelandic-constitution-crowdsourcing/ [Accedido el 27 de junio de 2020]

[9] Cortés-Cediel, M. E., Cantador, I., Fernández, M. (2020) "Exploiting Open Data to analyze discussion and controversy in online citizen participation"

[10] Cortés-Cediel, M. E., Cantador, I., & Bolívar, M. P. R. (2019). Analyzing Citizen Participation and Engagement in European Smart Cities. *Social Science Computer Review*. https://doi.org/10.1177/0894439319877478

[11] de Sousa Santos, B. (1998). Participatory Budgeting in Porto Alegre: Toward a Redistributive Democracy. Politics & Society, 26(4), 461–510. https://doi.org/10.1177/0032329298026004003

[12] Diccionario del español jurídico, (2020). "participación ciudadana" [online] https://dej.rae.es/lema/participaci%C3%B3n-ciudadana [Accedido el 27 de junio de 2020]

[13] Diccionario del español jurídico, (2020). "participación ciudadana" [online] https://dej.rae.es/lema/participaci%C3%B3n-ciudadana [Accedido el 27 de junio de 2020]

[14] Diccionario del español jurídico, (2020). "participación ciudadana" [online] https://dej.rae.es/lema/participaci%C3%B3n-ciudadana [Accedido el 27 de junio de 2020]

[15] Google Cloud, (2020). "Documentación de Dialogflow", Google Cloud. [online] https://cloud.google.com/dialogflow/docs/ [Accedido 01 de julio de 2020]

[16] Google, LLC, (2020). "Google Lens" [online] https://play.google.com/store/apps/details?id=com.google.ar.lens&hl=es [Accedido el 28 de junio de 2020]

[17] Google, LLC, (2020). "Te presentamos a tu Asistente de Google" [online] https://assistant.google.com/intl/es_es/ [Accedido el 28 de junio de 2020]

[18] Habib Olawale, W., (2018). “5 Benefits of using a Chatbot”, Medium. [online] https://chatbotslife.com/5-benefits-of-using-a-chatbot-937b8b793826 [Accedido el 30 de junio de 2020]
[19] Hernández Á. (2018) “Asistentes virtuales y chatbots: ¿mismo concepto?”, La Razón. [online] https://innovadores.larazon.es/es/asistentes-virtuales-y-chatbots-mismo-concepto/ [Accedido el 27 de junio de 2020]
[20] Imrie-Situnayake, D, (2018). “Basics of Dialogflow”, Dialogflow. [online] https://www.youtube.com/watch?v=9aHusGxntPw&list=PListM28kf0CNiMdmz-Ta5UcLR-ePCOW__ [Accedido 01 de junio de 2020]
[21] Information Retrieval Group, (2019). “Citizen participation in electronic participatory budgeting”, UAM. [online] http://ir.ii.uam.es/egov/ [Accedido el 30 de junio de 2020]
[22] Information Retrieval Group, (2019). “Citizen participation in electronic participatory budgeting”, UAM. [online] http://ir.ii.uam.es/egov/ [Accedido el 30 de junio de 2020]
[23] Li, A., (2019). “Google Assistant tests sending texts directly from the Android lockscreen”, 9to5Google. [online] https://9to5google.com/2019/07/21/google-assistant-lockscreen-texting/ [Accedido el 30 de junio de 2020]
[24] Lorda, T., (2018). “Casos de éxito de participación ciudadana por un mundo más justo y sostenible”, El independiente [online] https://www.elindependiente.com/desarrollo-sostenible/2018/09/24/casos-de-exito-de-participacion-ciudadana-por-un-mundo-mas-justo-y-sostenible/ [Accedido el 27 de junio de 2020]
[25] Lvovna, V., Votto, D., (2018). “What if Citizens Set City Budgets? An Experiment That Captivated the World—Participatory Budgeting—Might Be Abandoned in its Birthplace”, World Recurses Institute. [online] https://www.wri.org/blog/2018/06/what-if-citizens-set-city-budgets-experiment-captivated-world-participatory-budgeting [Accedido el 01 de julio de 2020]
[26] Microsoft Corporation (2020). “¿Qué es Cortana?” [online] https://support.microsoft.com/es-es/help/17214/cortana-what-is [Accedido el 28 de junio de 2020]
[27] Ministerio de Política Territorial y Función Pública, (2020). “portal de transparencia” [online] https://transparencia.gob.es/ [Accedido el 27 de junio de 2020]
[28] New York City Council, (2020). “New York City Council Participatory Budgeting” [online] http://ideas.pbnyc.org/page/about [Accedido el 27 de junio de 2020]
[29] NTS, (2020). “Diez ejemplos de cómo usar un 'chatbot' para mejorar tu negocio”, NTS. [online] https://www.nts-solutions.com/blog/chatbots-ejemplos.html [Accedido el 30 de junio de 2020]
[30] OECD (2001), Citizens as Partners: OECD Handbook on Information, Consultation and Public Participation in Policy-Making, OECD Publishing, Paris, https://doi.org/10.1787/9789264195578-en.
[31] PB Network, (2019). “Dundee Decides: Largest single PB in the UK in 2018” [online] https://pbnetwork.org.uk/dundee-decides-largest-single-pb-in-the-uk-in-2018/ [Accedido el 27 de junio de 2020]
[32] Planeta chatbot (2017). “¿Qué tipos de chatbots existen?”, Medium. [online] https://planetachatbot.com/tipos-de-chatbots-40682128324 [Accedido el 28 de junio de 2020]
[33] Smutny, P., Schreiberova, P., (2020). “Chatbots for learning: A review of educational chatbots for the Facebook Messenger” https://doi.org/10.1016/j.compedu.2020.103862

[34] Surmenok, P., (2016). “Chatbot Architecture”, Medium. [online] https://medium.com/@surmenok/chatbot-architecture-496f5bf820ed [Accedido el 27 de junio de 2020]
[35] Synpulse, (2017). “What Are Chatbots? Why Is Everybody Talking About Them?” synpulse https://www.synpulse.com/en_f/publications/article-en/get-ready-for-chatbots-part-1
[36] The Participatory Budgeting Project, (2020). “What is PB?” [online] https://www.participatorybudgeting.org/what-is-pb/ [Accedido el 27 de junio de 2020]
[37] Quesada Moreno, J. F., Callejas Carrión Z., Griol Barres, D. (2019) “Informe sobre sistemas conversacionales multimodales multilingües: tecnologías y arquitecturas para el Desarrollo de Asistentes Virtuales, Sistemas de Diálogo y Otros Interfaces Conversacionales” [online] https://www.plantl.gob.es/tecnologias-lenguaje/actividades/Estudios%20tcnicos%20y%20de%20gobernanza/Estudio%20de%20sistemas%20conversacionales/estudio-sistemas-conversacionales.pdf [Accedido el 27 de junio de 2020]
[38] Vergadia, P., (2019) “Deconstructing Chatbots”, Google Cloud Plataform. [online] https://youtu.be/O00K10xP5MU [Accedido 01 de julio de 2020]
[39] VMWare, Inc., (2020). “Spring | Guides” [online] https://spring.io/guides [Accedido el 01 de julio de 2020]
[40] Yúbal FM (2018). “Qué es Alexa, qué puedes hacer con él y qué dispositivos son compatibles” Xataka Basics. [online] https://www.xataka.com/basics/que-alexa-que-puedes-hacer-que-dispositivos-compatibles [Accedido 28 de junio de 2020]
[41] Zheng, Y., Schachter, H.L. (2017). “Explaining Citizens’ E-Participation Use: The Role of Perceived Advantages”. Public Organiz Rev 17, 409–428 (2017). https://doi.org/10.1007/s11115-016-0346-2

Glosario

Agente conversacional	Tipo de sistema conversacional cuya funcionalidad es lo por lo general más acotada y rígida que la de un asistente personal.
Asistente personal	Tipo de sistema conversacional cuya funcionalidad es lo suficientemente amplia como para actuar como una especie de secretario virtual
Decide Madrid	Web de participación ciudadana del ayuntamiento de Madrid donde la ciudadanía puede proponer, apoyar o debatir acerca de propuestas o medidas políticas.
Dialogflow	Plataforma desarrollada por Google, LLC utilizada para el diseño y la integración de un agente conversacional.
Entity	Parámetros de un agente Dialogflow compuesto por un listado de palabras clave y sus sinónimos.
Intent	Clasificador de intención de usuario de un agente Dialogflow compuesto por frases de entrenamiento, parámetros, contextos y respuestas por defecto.
Maven	Herramienta software para la estandariza la configuración y construcción un proyecto Java.
PostgreSQL	Sistema de gestión de bases de datos relacionales con orientación a objetos y distribuido en formato de código abierto.
Presupuesto participativo	Procesos democráticos donde la comunidad decide democráticamente el uso de parte del presupuesto.
Sistema conversacional	Sistemas informáticos cuya interfaz se basa en un diálogo a través de los cual los usuarios pueden comunicarse en tiempo real mediante el uso de lengua natural, ya sea escrito o hablado.
Spring Boot	Infraestructura ligera que facilita la configuración y ejecución de aplicaciones Spring.
Telegram	Plataforma de mensajería y VOIP dirigida a la mensajería instantánea donde se integró el agente de Dialogflow.

Anexos

A Flujo de conversación

Durante la fase de diseño se realizó un diagrama de flujo simplificado que delimita las posibles direcciones que podían la conversación, procurando contemplar todos los posibles cambios de intencionalidad del usuario.

Para empezar, tenemos el siguiente diagrama simplificado que muestra el flujo de los posibles intents.

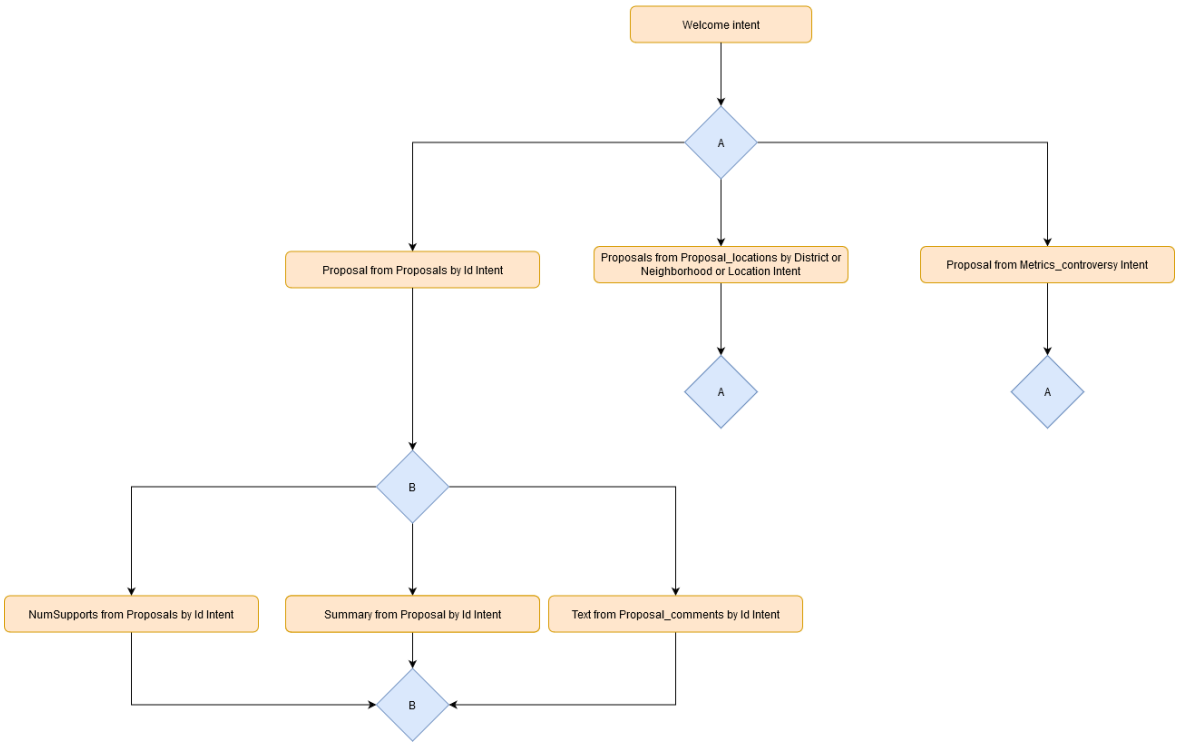

Figura 0-1: Diagrama de flujo general

El flujo de la conversación iniciará siempre con un saludo por parte del agente, siempre y cuando no se reste reanudando una conversación previa.

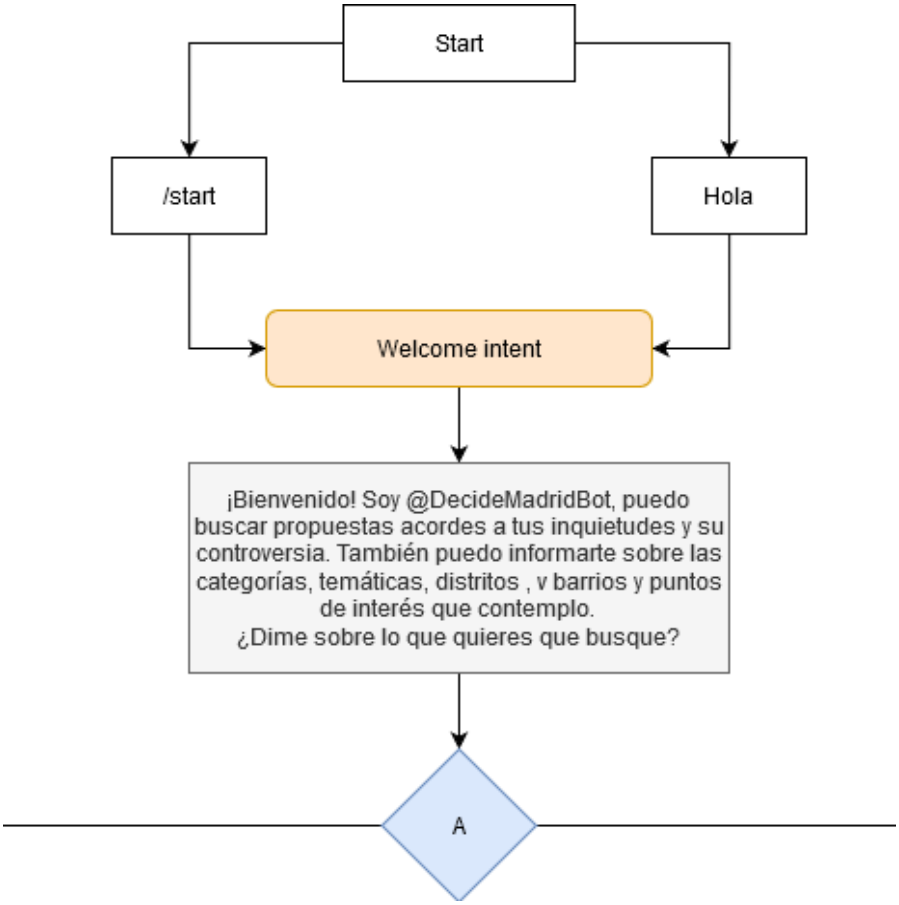

Figura 0-2: Flujo Welcome

Una vez finalizado el saludo, el usuario retoma el control y puede llevar la conversación para la intención que quiera (salvo la de ver número de apoyos, resumen y comentarios de una propuesta, puesto que no ha iniciado ningún contexto). Para simplificar el diagrama se han retirado todos los contextos de listados auxiliares (categorías, temática, distritos, barrios y localizaciones) quedándose con los flujos más representativos de la aplicación integrados en la última iteración.

El primer que se va a estudiar en la búsqueda de propuestas por un filtro, en particular por un filtro de geolocalización, por lo que el filtro puede ser tanto un distrito, un barrio o un punto de interés, mostrando un listado con el título y el identificador de cada propuesta que cumple los requisitos. En el caso de que el filtro fuera una categoría o una temática, el esquema sería equivalente cambiando tan solo el intent y sustituir la llamada interna que se hace al método *processProposalLocations* por *processProposalCategoriesAndTopics.*

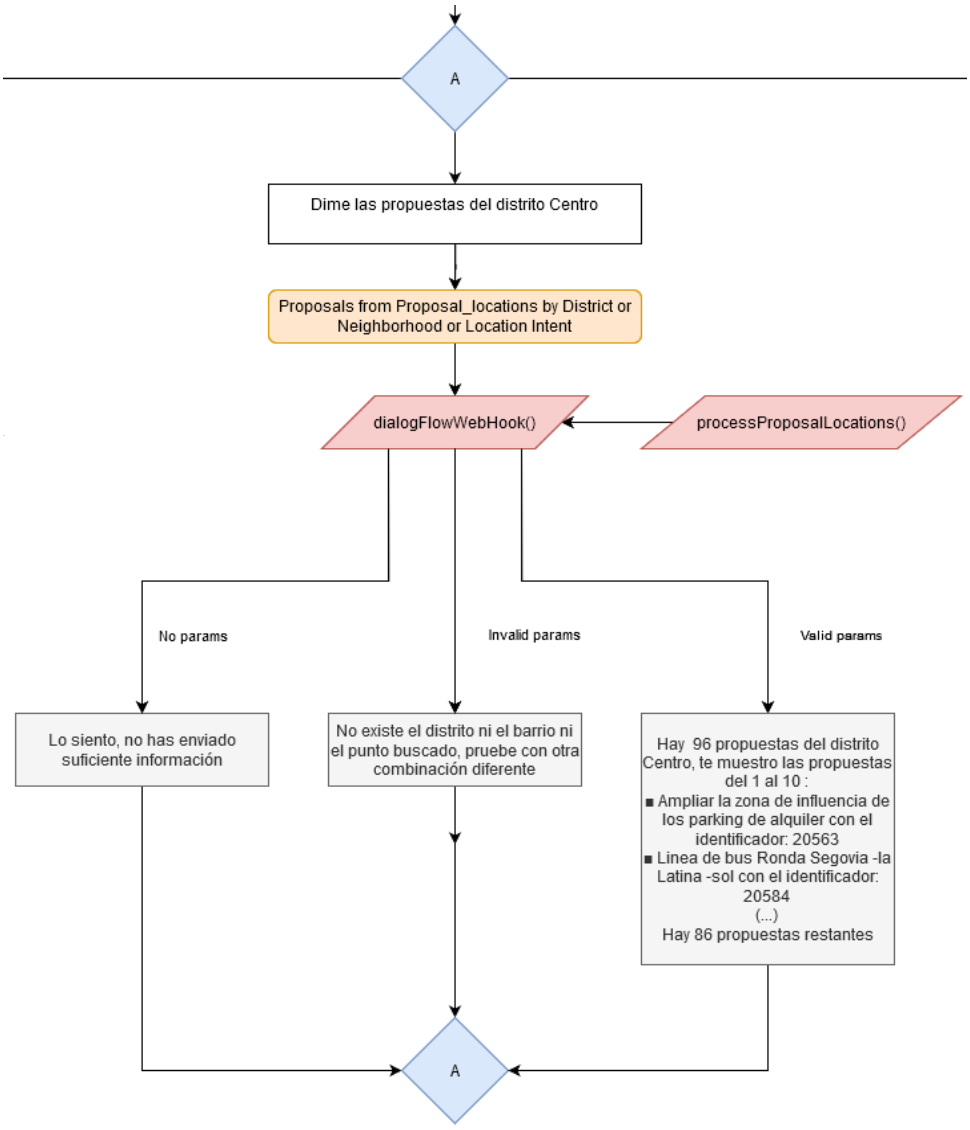

Figura 0-3: Flujo Proposals from Proposal_locations

El siguiente flujo que se profundizará es la búsqueda de una propuesta particular por medio de su identificador, el cual se desarrolla como se aprecia en la imagen inferior. Tal y como se comentó en la sección de desarrollo, la peculiaridad de este intent es que inicia un contexto que almacena el identificador de la propuesta búsqueda (en el caso de una pregunta afirmativa.

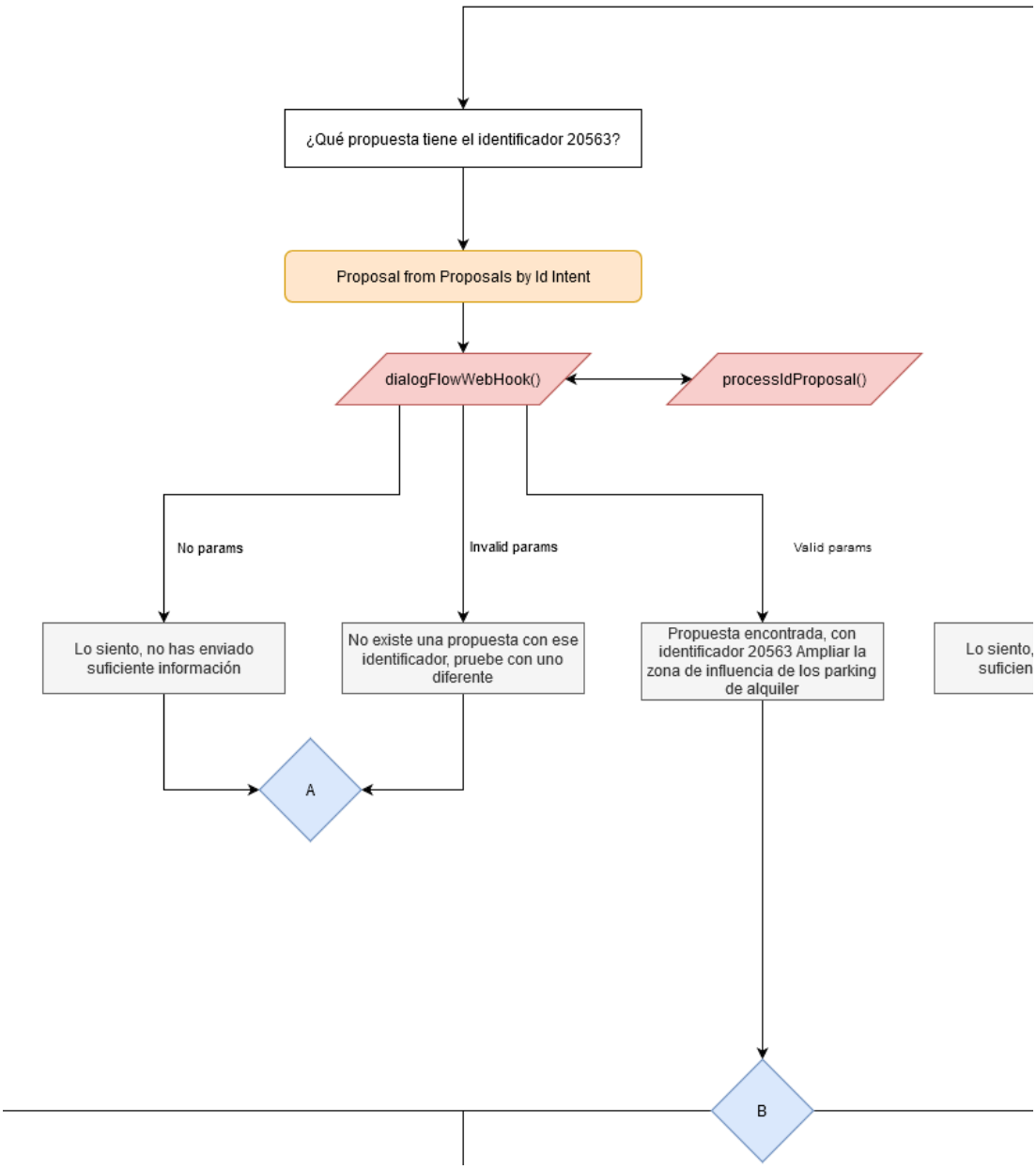

Figura 0-4: Flujo Proposal from Proposal by Id

Los siguientes tres flujos solamente se pueden contemplar si existe el contexto previo del identificador (B), ya que de lo contrario no identificará el intent como tal. El primero de estos flujos es el de número de apoyos que tiene una propuesta del contexto, el intent continúa conservando el contexto para posible futuros flujos acerca de la propuesta.

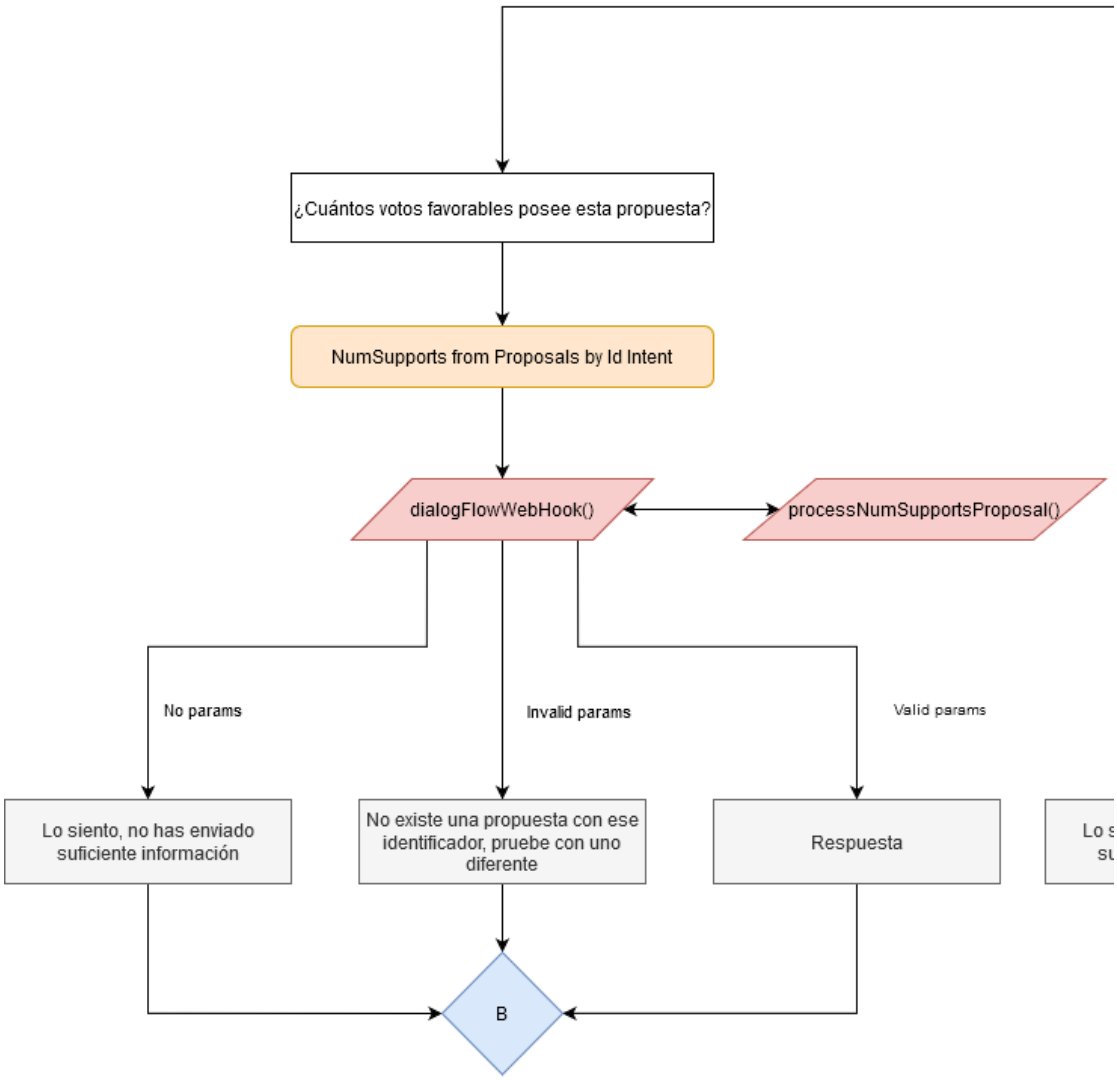

Figura 0-5: Flujo NumSupports from Proposal by Id

El segundo es el resumen de la propuesta del contexto, el agente muestra un pequeño fragmento de texto que condensa toda la propuesta y adjunta un enlace de referencia a Decide Madrid. Sigue mantenido en el contexto la información del identificador por si se continúan haciendo preguntas acerca la propuesta. Se sigue manteniendo en el contexto la información del identificador por si se continúan haciendo preguntas acerca la propuesta.

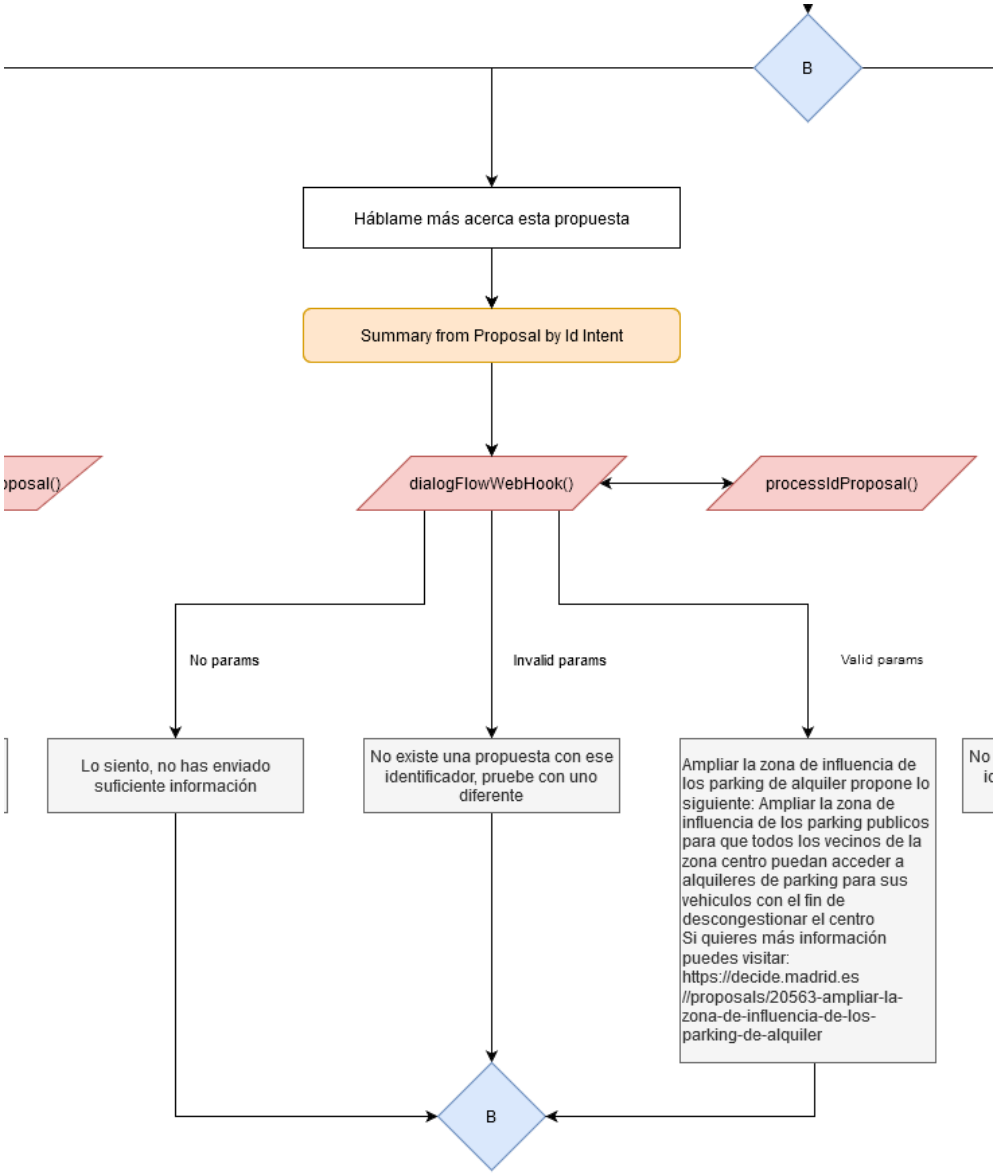

Figura 0-6: Flujo Summary from Proposal by Id

El tercero es el listado de comentarios de la propuesta del contexto. El mensaje muestra el usuario que realizo el mensaje, si se trata de una respuesta a otro usuario, el propio texto del comentario y la fecha en la que se realizó. Al igual que en los dos flujos previos se continúa conservando el identificador de la propuesta en el contexto.

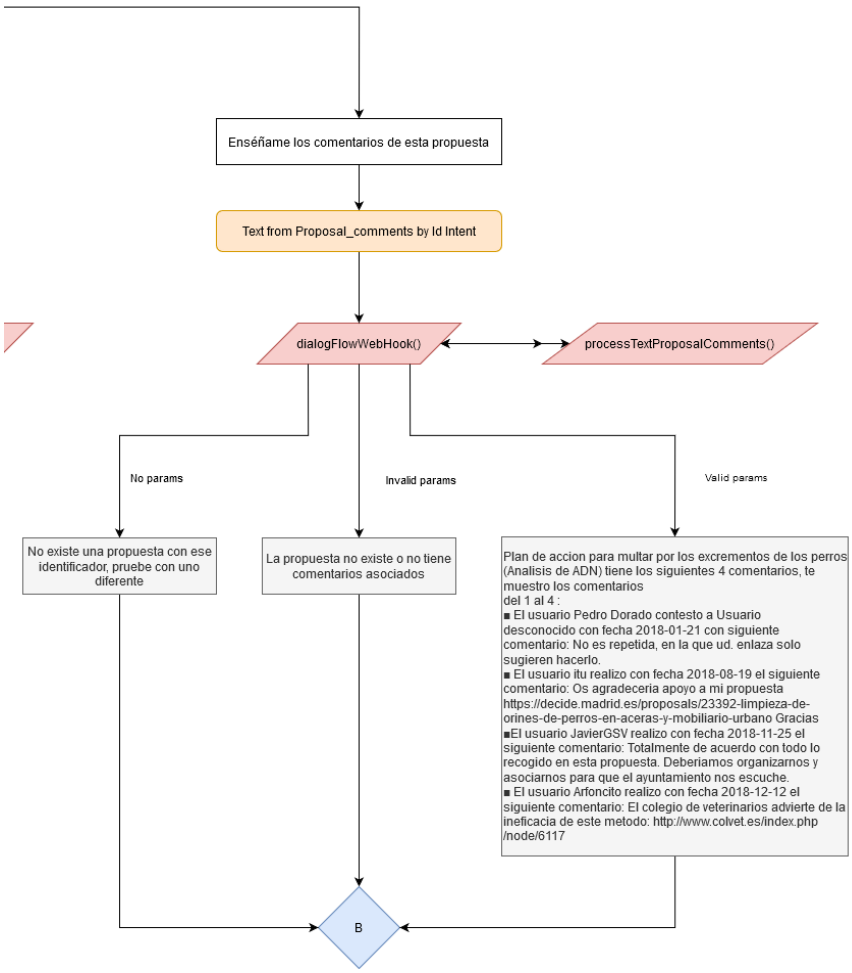

Figura 0-7: Flujo Text from Proposal_comments by Id

Volviendo al flujo inicial(A), se procede a analizar el último flujo importante contemplado, la búsqueda de propuestas según su nivel de controversia. Este flujo admite cualquier tipo de filtro (categoría, temática, distrito, barrio o punto de interés) y muestra las propuestas ordenadas de mayor a menor o viceversa las propuestas más controvertida (todas o las propuestas que cumplan el filtro adjunto)

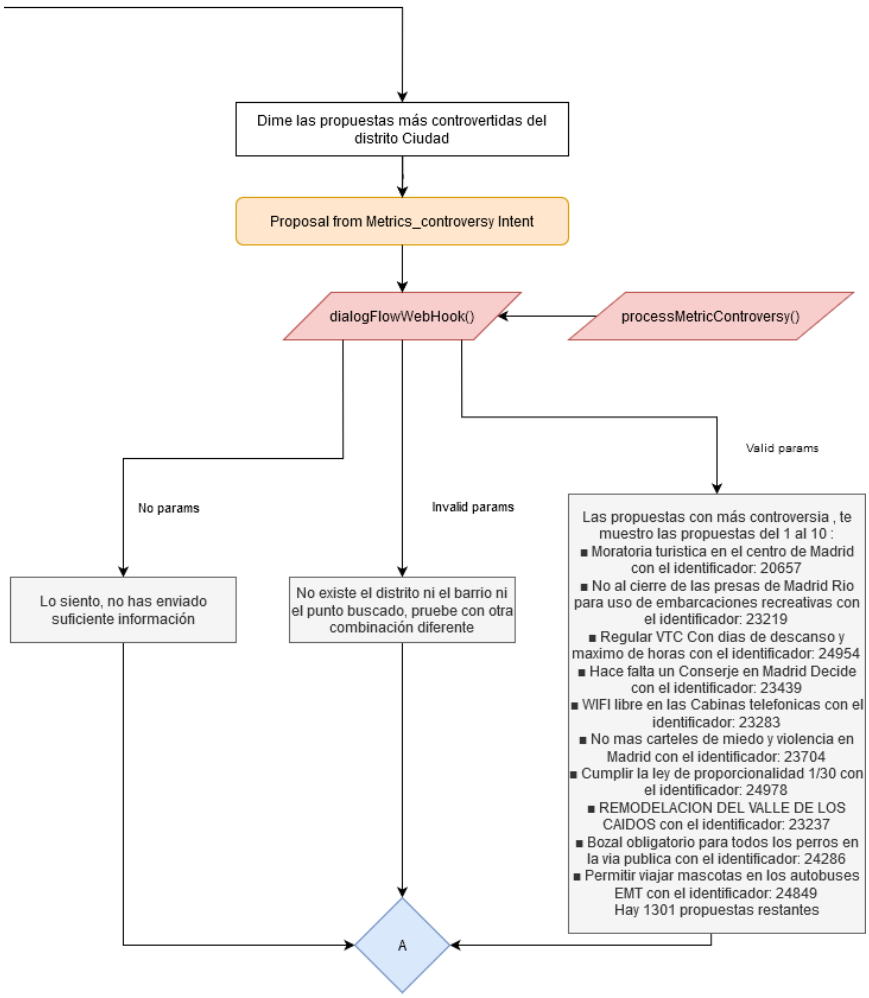

Figura 0-8: Proposal from Metrics_controversy

A parte de estos flujos están los flujos que realizan listados de categorías, temáticas, distritos, barrios o puntos de interés que siguen prácticamente el mismo esquema que los flujos vistos, no se han añadido con el propósito de clarificar el diagrama de flujo y no saturas este anexo.

B Configuración de servicio web y agent

En la siguiente sección se mostrará la configuración de los ficheros del servicio web y de las entidades del agente conversacional.

pom.xml

Una de las mayores ventajas Spring Boot de Java se basan en la simplificación tanto de la configuración como del desarrollo. En esta línea una de las características más destacables es el Project Object Models, abreviado como POM, que nos permite simplificar la configuración de Maven a un único fichero xml.

Este fichero tiene dos partes fundamentales, por un lado, están las dependencias que son lo que permiten a Spring Boot buscar los jar asociados e importarlos a la aplicación.
En este caso tenemos tres grupos, el primero son las dependencias de Spring Boot, la cual nos permiten habilitar el núcleo de la aplicación, las dependencias de jpa y postgresql que habilitan la conexión con una base de datos y la de Google que nos permite establecer la conexión con el agente de Dialogflow.

```
21  <dependencies>
22      <dependency>
23          <groupId>org.springframework.boot</groupId>
24          <artifactId>spring-boot-starter-cloud-connectors</artifactId>
25      </dependency>
26      <dependency>
27          <groupId>org.springframework.boot</groupId>
28          <artifactId>spring-boot-starter-data-jpa</artifactId>
29      </dependency>
30      <dependency>
31          <groupId>org.springframework.boot</groupId>
32          <artifactId>spring-boot-starter-web</artifactId>
33      </dependency>
34
35      <dependency>
36          <groupId>org.postgresql</groupId>
37          <artifactId>postgresql</artifactId>
38          <scope>runtime</scope>
39      </dependency>
40      <dependency>
41          <groupId>org.springframework.boot</groupId>
42          <artifactId>spring-boot-starter-test</artifactId>
43          <scope>test</scope>
44          <exclusions>
45              <exclusion>
46                  <groupId>org.junit.vintage</groupId>
47                  <artifactId>junit-vintage-engine</artifactId>
48              </exclusion>
49          </exclusions>
50      </dependency>
51
52      <!-- Google -->
53      <dependency>
54          <groupId>com.google.apis</groupId>
55          <artifactId>google-api-services-dialogflow</artifactId>
56          <version>v2beta1-rev15-1.23.0</version>
57      </dependency>
58  </dependencies>
```

Figura 0-9: Dependecies del pom.xml

Por otro lado, tenemos los plugins los cuales nos permiten invocar complementos.

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Figura 0-10: Plugins del pom.xml

Application.properties

Este fichero nos permite establecer la configuración con la base de datos alojada en local, en primer lugar, comunicamos que se a tratar de la plataforma PostgreSQL, en el segundo lugar le indicamos la ubicación de la base de datos que en nuestro caso será en local en el puerto 5432 con el nombre de decide.madrid.tfg. Los dos siguientes líneas son el usuario y la contraseña del creador de loa base de datos y por último, el hibernate establece el funcionamiento de la base de datos, por ejemplo update se limitará a actualizar los campos de una base ya existente, créate se limitará a crear una base de datos cada vez que se inicie el servicio y create-update se limitará a crear la base de datos y a permitir actualizarla cada hasta que se pare el servicio.

```
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/decide.madrid.tfg
spring.datasource.username=postgres
spring.datasource.password=admin

spring.jpa.hibernate.ddl-auto=update
```

Figura 0-11: Application.properties

Entities.csv

A continuación, mostramos algunos fragmentos del CSV que conforma las entidades del agente, en este tipo de ficheros, la primera columna determina el valor que toma el filtro y el resto son los sinónimos asociados a ese valor.

```
"Accesibilidad","Accesibilidad","accesibilidad"
"Animales","Animales","animales"
"Asociaciones","Asociaciones","asociaciones","Asociacion","asociacion"
"Civismo","Civismo","civismo"
"Cultura","Cultura","cultura"
"Delicuencia","Delicuencia","delicuencia"
"Deportes","Deportes","deportes"
"Derechos sociales","Derechos sociales","derechos sociales"
"Empleo","Empleo","empleo"
"Familia e infancia","Familia e infancia","familia e infancia"
"Justicia","Justicia","justicia"
"Medio ambiente","Medio ambiente","medio ambiente"
"Movilidad","Movilidad","movilidad"
"Ocio y entretenimiento","Ocio y entretenimiento","ocio y entretenimiento"
"Salud y sanidad","Salud","salud"
"Seguridad y emergencias","Seguridad y emergencias","seguridad y emergencias"
"Sostenibilidad","Sostenibilidad","sostenibilidad"
"Tercera edad","Tercera edad","tercera edad"
"Transparencia","Transparencia","transparencia"
"Turismo","Turismo","turismo"
"Urbanismo","Urbanismo","urbanismo"
"Vivienda","Vivienda","vivienda"
```

Figura 0-12 : Categories.csv

```
"abono de transportes", "abono de transportes", "abono de transporte", "abonos de trans
"accesibilidad peatonal", "accesibilidad peatonal"
"actividad economica", "actividad economica"
"actividad fisica", "actividad fisica"
"adiestramiento canino", "adiestramiento canino"
"adolescencia", "adolescencia"
"alergias", "alergias"
"alquiler de vivienda", "alquiler de vivienda"
"alumbrado", "alumbrado"
"ambulancias", "ambulancias"
"animales callejeros", "animales callejeros"
"aparatos de gimnasia", "aparatos de gimnasia"
"aparcamiento regulado", "aparcamiento regulado"
"aparcamientos", "aparcamientos"
"aparcamientos para discapacitados", "aparcamientos para discapacitados"
"arboles", "arboles"
"areas caninas", "areas caninas"
"arte urbano", "arte urbano"
"artistas", "artistas"
"asociacionismo", "asociacionismo"
"atascos", "atascos"
"atletismo", "atletismo"
"atracos", "atracos", "atraco", "Atraco", "Atracos", "hurto", "hurtos", "Hurtos"
"autobuses", "autobuses", "autobus", "Autobuses", "Autobus"
"autobuses nocturnos", "autobuses nocturnos"
"autobuses turisticos", "autobuses turisticos"
"ayudas economicas", "ayudas economicas"
"ayudas sociales", "ayudas sociales"
"ayuntamiento", "ayuntamiento"
"baches", "baches"
"baile", "baile"
"baloncesto", "baloncesto"
```

Figura 0-13: Topics.csv

```
"Retiro", "Retiro"
"Latina", "Latina"
"Villaverde", "Villaverde"
"Puente de Vallecas", "Puente de Vallecas"
"Salamanca", "Salamanca"
"Villa de Vallecas", "Villa de Vallecas"
"Centro", "Centro"
"Barajas", "Barajas"
"Arganzuela", "Arganzuela"
"Ciudad", "Ciudad"
"Carabanchel", "Carabanchel"
"Chamartín", "Chamartín"
"Ciudad Lineal", "Ciudad Lineal"
"Moratalaz", "Moratalaz"
"Chamberí", "Chamberí"
"Hortaleza", "Hortaleza"
"Tetuán", "Tetuán"
"San Blas-Canillejas", "San Blas-Canillejas"
"Moncloa-Aravaca", "Moncloa-Aravaca"
"Usera", "Usera"
"Vicálvaro", "Vicálvaro"
"Fuencarral-El Pardo", "Fuencarral-El Pardo"
```

Figura 0-14: Districts.csv

```
"Aravaca", "Aravaca"
"Pueblo Nuevo", "Pueblo Nuevo"
"Jerónimos", "Jerónimos"
"Orcasur", "Orcasur"
"Guindalera","Guindalera"
"Palomas", "Palomas"
"Casco Histórico de Vicálvaro", "Casco Histórico de Vicálvaro"
"Castilla", "Castilla"
"San Andrés", "San Andrés"
"Los Cármenes", "Los Cármenes"
"Ciudad Jardín", "Ciudad Jardín"
"Ensanche de Vallecas", "Ensanche de Vallecas"
"Berruguete", "Berruguete"
"Castellana", "Castellana"
"Valverde", "Valverde"
"Salvador", "Salvador"
"Pradolongo","Pradolongo"
"Piovera","Piovera"
"Valdemarín", "Valdemarín"
"Palacio", "Palacio"
"Los Rosales", "Los Rosales"
"Las Águilas", "Las Águilas"
"Cuatro Vientos", "Cuatro Vientos"
"Apóstol Santiago", "Apóstol Santiago"
"Bellas Vistas", "Bellas Vistas"
"Delicias", "Delicias"
"Niño Jesús", "Niño Jesús"
"Legazpi", "Legazpi"
"Concepción", "Concepción"
```

Figura 0-15. Neighborhoods.csv

```
"chueca", "chueca", "Chueca"
"avenida de pablo neruda", "avenida de pablo neruda"
"calle de samaria", "calle de samaria"
"parque de la bombilla", "parque de la bombilla"
"plaza mayor", "plaza mayor"
"avenida de niza", "avenida de niza"
"paseo del prado", "paseo del prado"
"calle de las sicelidas", "calle de las sicelidas"
"avenida de valladolid", "avenida de valladolid"
"plaza de la republica argentina", "plaza de la republica argentina"
"avenida de san luis", "avenida de san luis"
"valdebebas", "valdebebas", "Valdebebas"
"calle fuencarral", "calle fuencarral"
"avenida de asturias", "avenida de asturias"
"plaza de oriente", "plaza de oriente"
"feria de madrid", "feria de madrid"
"museo del ferrocarril", "museo del ferrocarril"
"plaza de san vicente de paul", "plaza de san vicente de paul"
"paseo de la chopera", "paseo de la chopera"
"jardines de sabatini", "jardines de sabatini"
"calle de la via", "calle de la via"
```

Figura 0-16: Locations.csv

C Conversación de ejemplo

En este apartado vamos a realizar una pequeña conversación de ejemplo con el objetivo de mostrar la funcionalidad del prototipo.

Comenzamos la conversación con el chatbot, esto puede ser mediante el comando /start o mediante un saludo, el agente escogerá uno de los saludos por defecto. A continuación, el usuario hace una pregunta sobre las propuestas disponibles, indicando el flujo de la figura 0-3, en este caso el agente al considerarlo un flujo válido devuelve un listado de las propuestas del distrito con su identificador asociado.

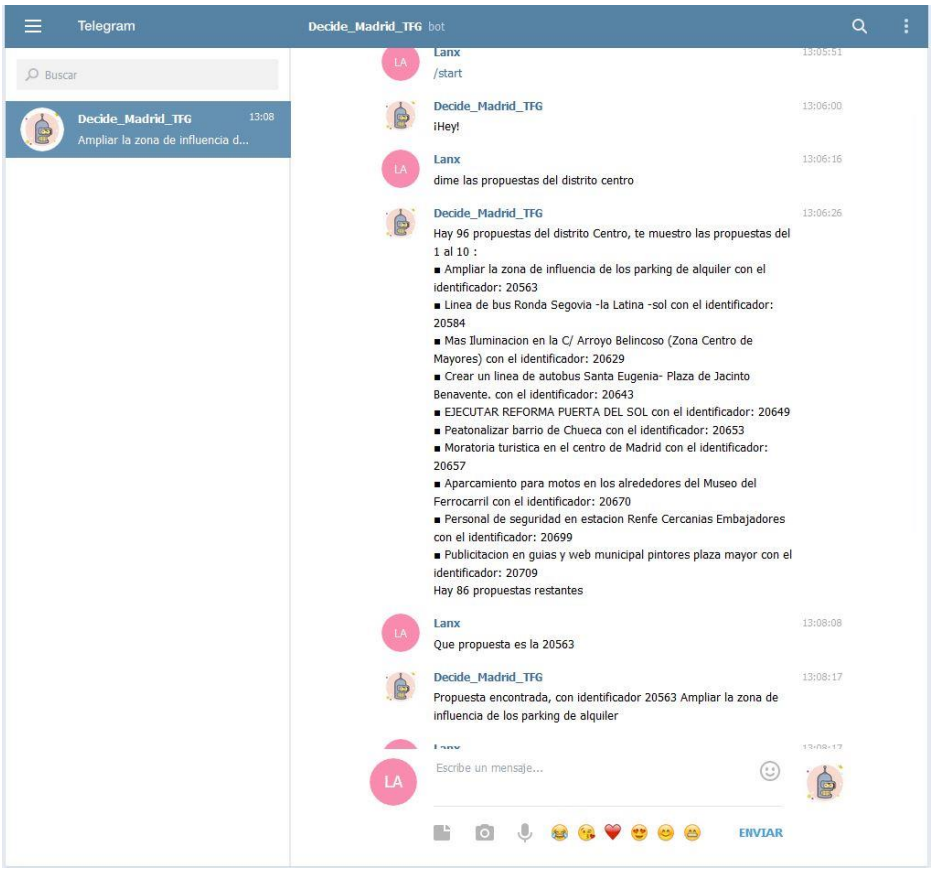

Figura 0-17: Saludo y búsqueda por distrito

Seguido a esto el usuario busca la primera propuesta por su identificador, iniciando el flujo 0-4 y guardando en su contexto el identificador de la propuesta, al existir la propuesta el agente devuelve el título y el identificador de la propuesta. Seguidamente el usuario pregunta por el detalle de la propuesta.

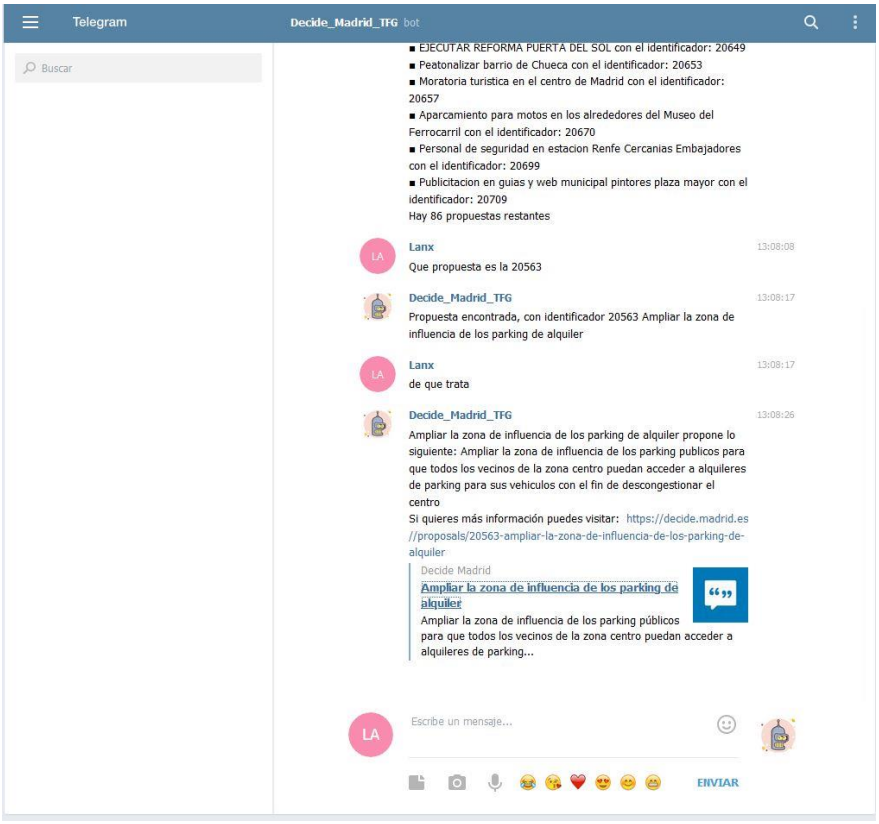

Figura 0-18: Propuesta por identificador y detalle de la propuesta

A continuación, volvemos a preguntar por propuestas, esta vez mediante un filtro de categoría. El agente detecta Animales como un filtro válido y muestra un listado de las propuestas con sus identificadores. El usuario entonces pregunta por una propuesta particular de listado, el agente muestra el título y el identificador de la propuesta y carga un nuevo contexto asociado.

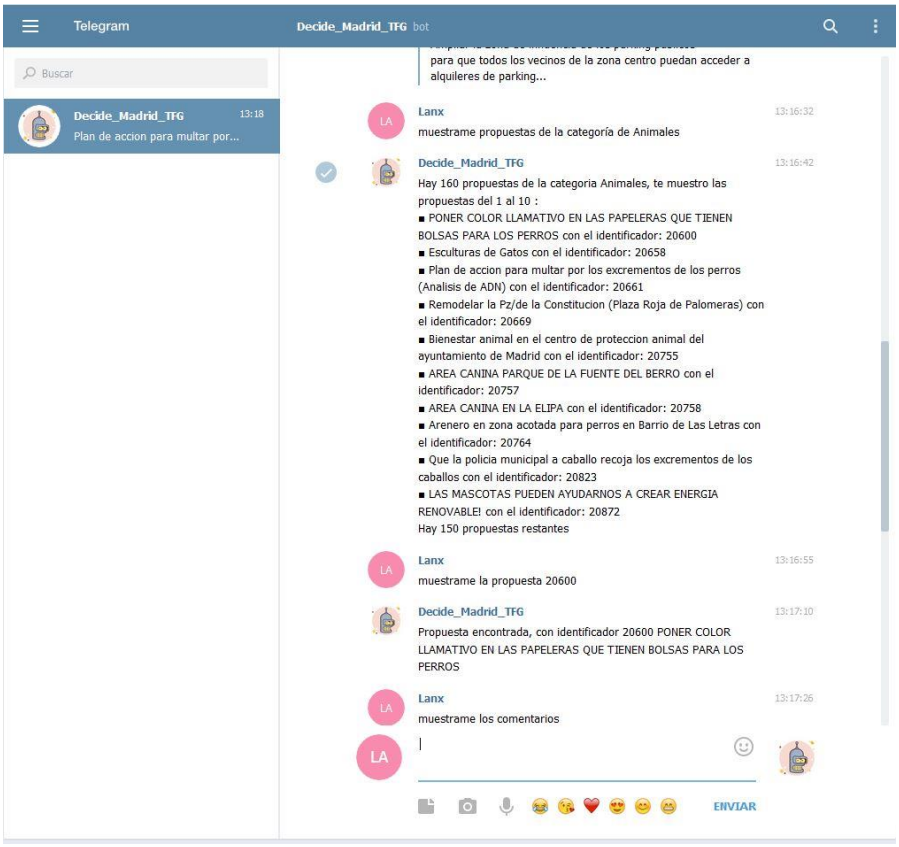

Figura 0-19: Búsqueda de propuesta por categoría

En este caso preguntamos por los comentarios de la propuesta, el agente nos comunica que efectivamente no hay comentarios asociados de la propuesta. Ahora el usuario busca por otra propuesta (que tenga comentarios) y carga un nuevo contexto, esta vez al preguntar por los comentarios el agente muestra un listado de los comentarios.

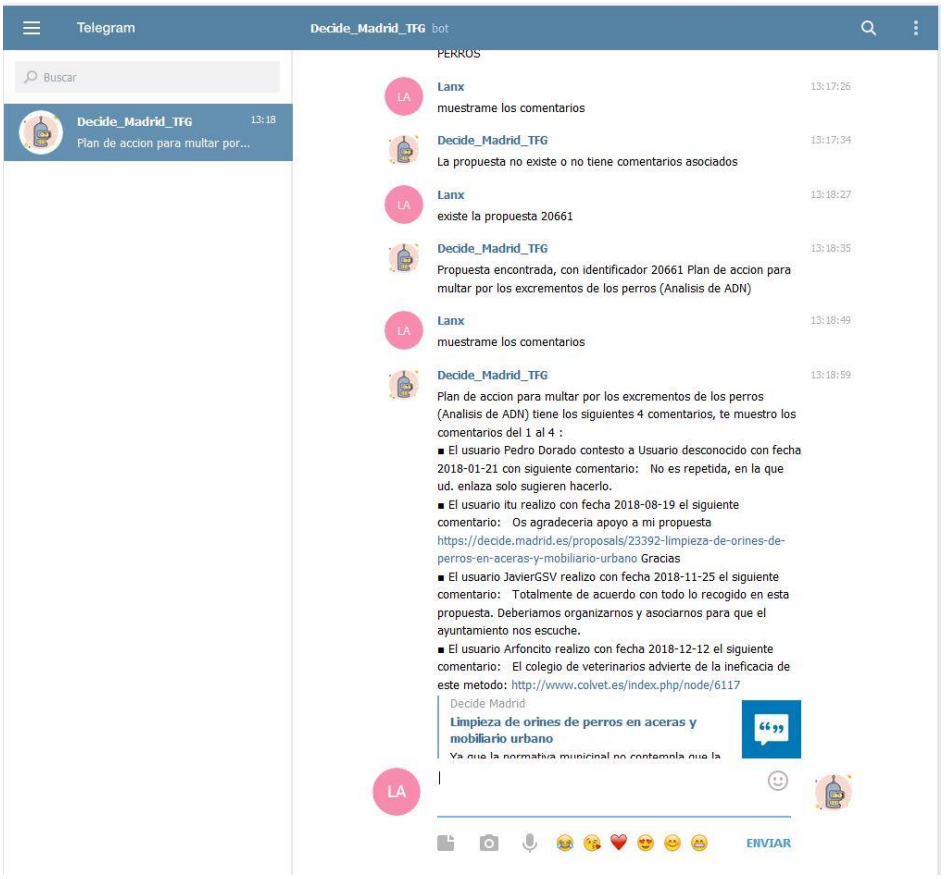

Figura 0-20: Listado de comentarios de propuesta asociada

A continuación, realizamos una búsqueda de propuestas por su nivel de controversia, además le incluimos el filtro de distritos. El agente lista las propuestas que cumplen el filtro del distrito ordenado los de mayor controversia a menor controversia

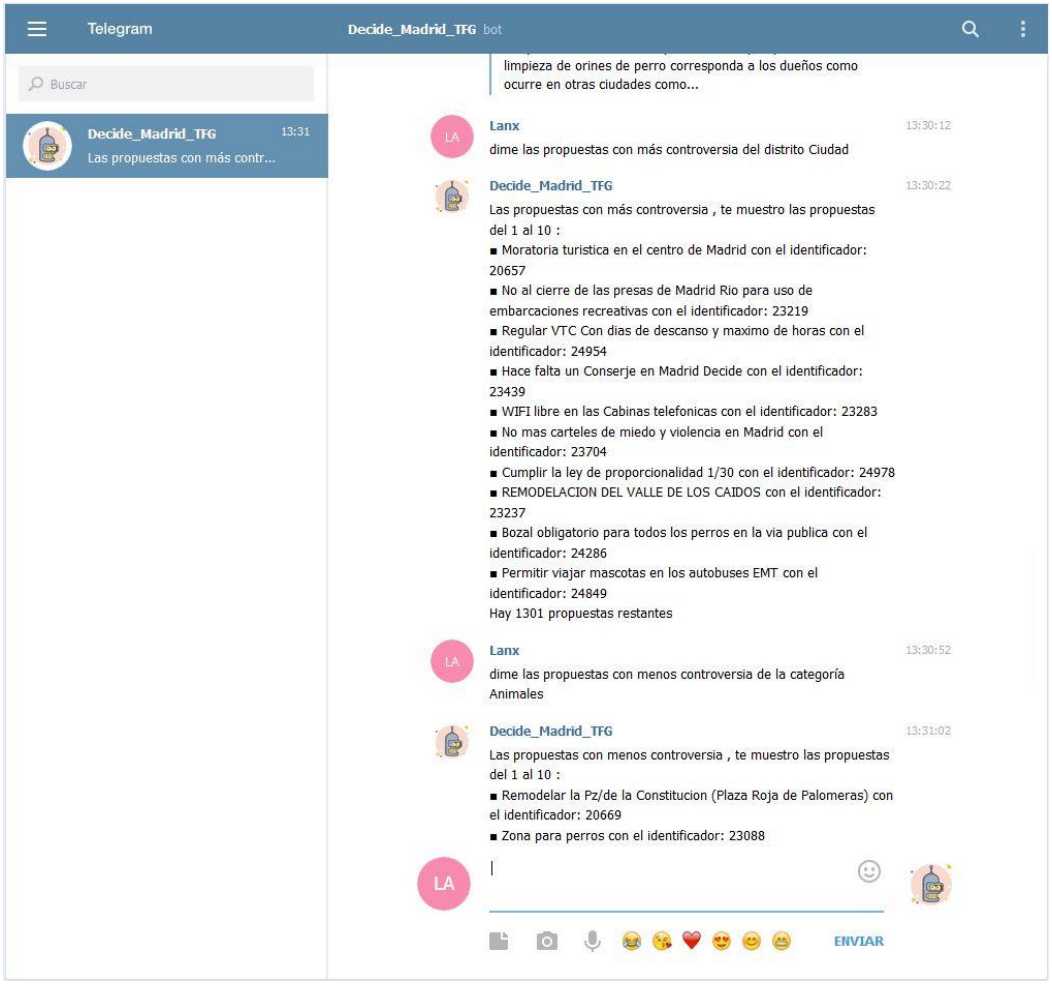

Figura 0-21: Búsqueda de propuestas por controversia y distritos

También podemos buscar las propuestas ordenándolos por menor nivel de controversia, tal como se ve en la figura inferior.

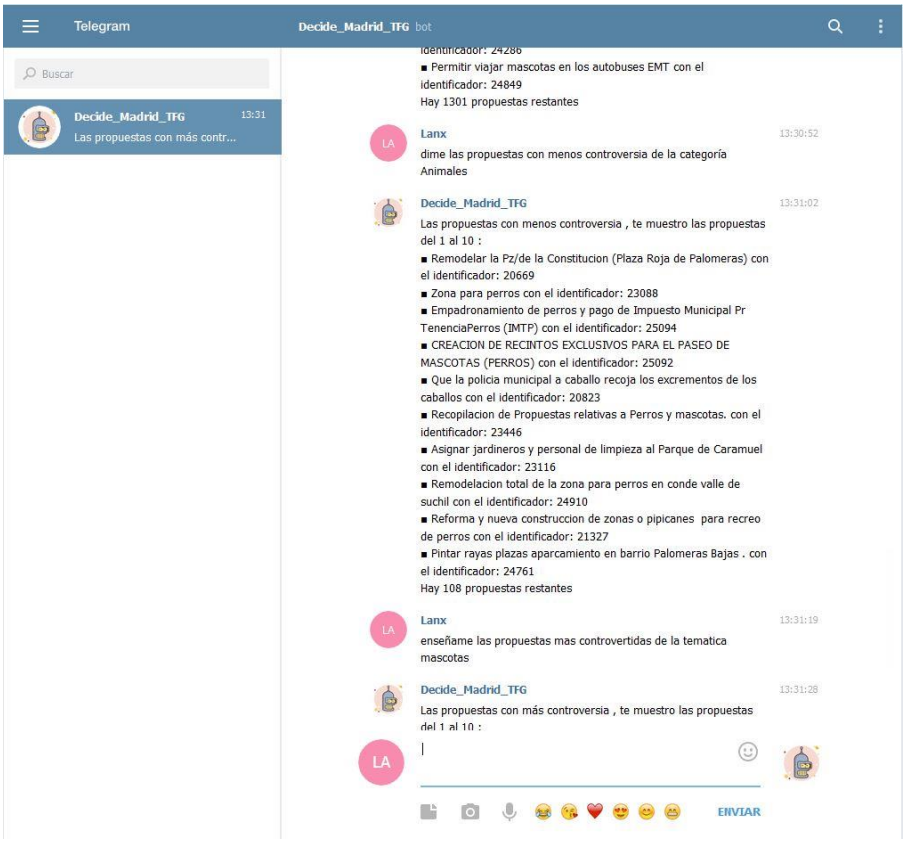

Figura 0-22: Búsqueda de propuestas por controversia y categoría

Aquí el usuario pide otra vez las propuestas ordenadas por controversia, con la salvedad de que esta vez utilizamos el filtro de temática. Por último, el usuario muestra un mensaje de despedida, por lo tanto, el agente lo detecta como fin de conversación y finaliza.

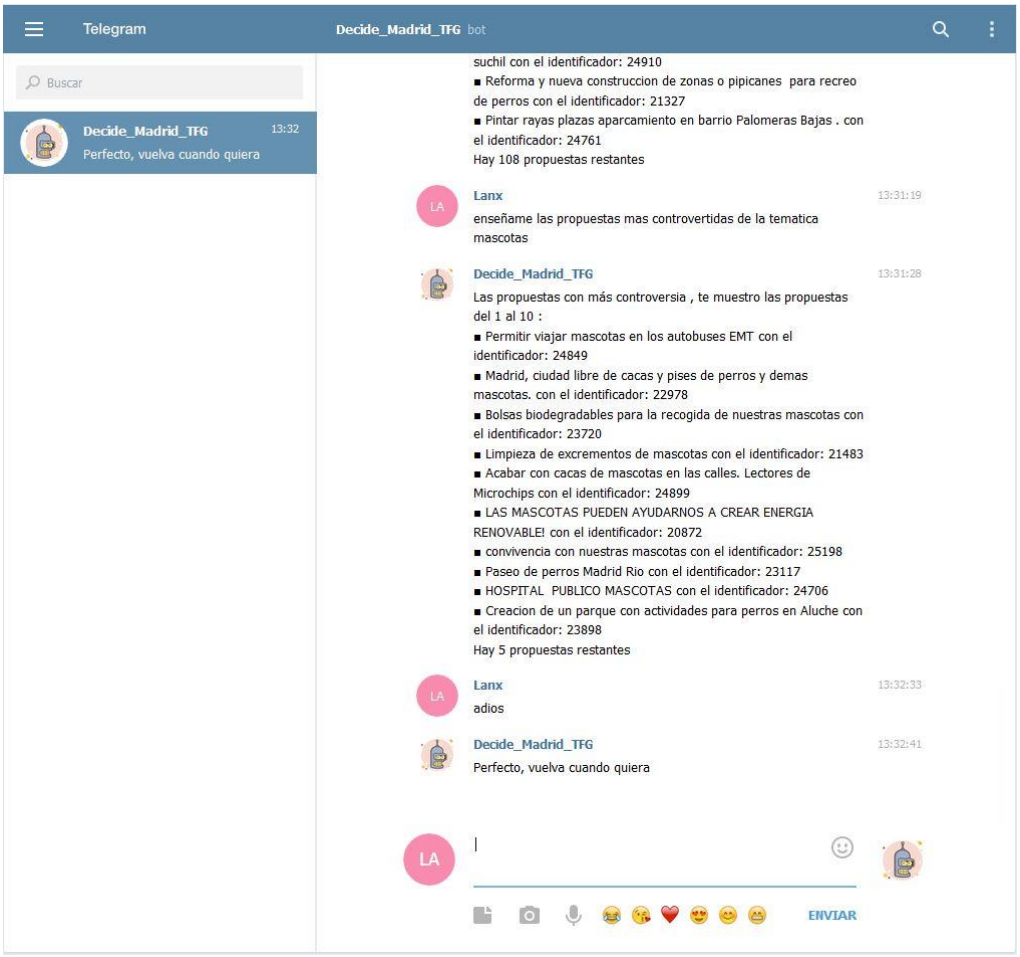

Figura 0-23: Búsqueda de propuestas por controversia y temática

Iniciando una nueva conversación, el usuario pide un listado de las categorías. El agente realiza un listado de todos los filtros disponibles, de igual forma cuando pregunta por temáticas el agente elabora un nuevo listado. Estos mismos listados se pueden hacer para el resto de los filtros: distritos, barrios o puntos de interés.

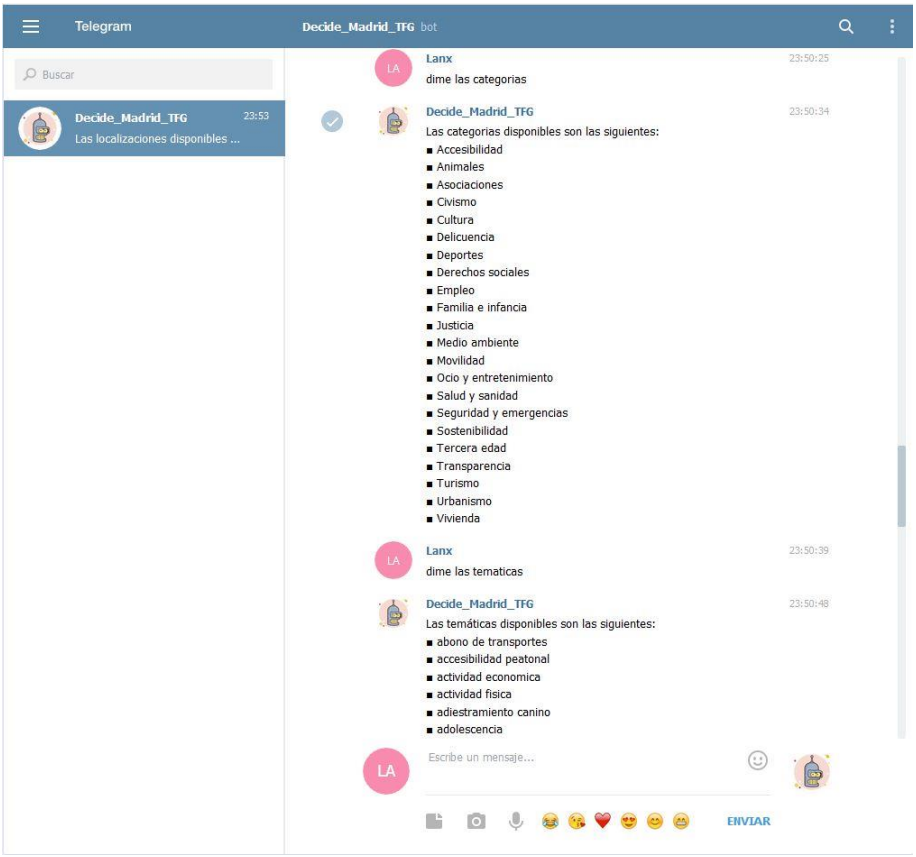

Figura 0-24: Listado de filtros disponibles